UCRL-LR-112583

Target Scheduling for Directed Energy Weapon Platforms

G. C. Corynen

January 1993

Work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

Accession Number: 4551

Publication Date: Jan 01, 1993

Title: Target Scheduling for Directed Energy Weapon Platforms

Personal Author: Corynen, G.C.

Corporate Author Or Publisher: Lawrence Livermore National Laboratory, Livermore, CA 94551 Report Number: UCRL-LR-112583

Descriptors, Keywords: Target Schedule Directed Energy Weapon DEW Boost Phase Platform Acquisition Track Laser HEL

Pages: 00171

Cataloged Date: Jul 06, 1993

Document Type: HC

Number of Copies In Library: 000001

Record ID: 27368

UCRL-LR-112583
Distribution Category UC-705

Target Scheduling for Directed Energy Weapon Platforms

G. C. Corynen

Manuscript date: January 1993

LAWRENCE LIVERMORE NATIONAL LABORATORY
University of California • Livermore, California • 94551

Contents

List of Figures

Nomenclature

ATP	Acquisition, Tracking, and Pointing
B&E	Background and Environment
C-language	object- and pointer-oriented scientific programming language with dynamic memory allocation capabilities
CDF	Cumulative Distribution Function
DDTS	Deadline-Driven Target Scheduling
DE	Directed Energy
DEW	Directed Energy Weapon
FB	forebody
FS	fast steering subsystem
HEL	high-energy laser
LLNL	Lawrence Livermore National Laboratory
NN	nearest neighbor
NPB	neutral particle beam
pdf	probability density function
RL	Rome Laboratory, Griffiss Air Force Base, Rome, NY
RV	Reentry Vehicle
SDI	Strategic Defense Initiative
SPT	shortest processing time
THREATSIM	a ballistic missile threat simulation package
TOL	accuracy tolerance variable in the BRENT algorithm
TSP	Target Scheduling Problem
WTA	Weapon-Target Assignment

Target Scheduling for Directed Energy Weapon Platforms

Executive Summary

Introduction

This final report documents the results of a three-year technology development program sponsored by the Rome Laboratory (RL) as part of the Strategic Defense Initiative (SDI) and executed at the Lawrence Livermore National Laboratory (LLNL). The major objectives of this program were to develop, test, and deliver algorithms for managing Directed Energy Weapons (DEW) platforms during defensive engagements with a number of offensive weapons, which we shall call the *targets*. The main focus of this program has been on space-based High-Energy Lasers (HEL) and Neutral Particle Beam (NPB) platforms operating in earth-orbit during the boost and midcourse phases.

The Real-Time Target Scheduling Problem

During the prosecution of targets, the effective allocation of weapon energy is the key to achieving optimal platform performance. This involves three major real-time tasks:

1. To derive the order in which targets should be prosecuted.
2. To determine how much energy and time should be allocated to each target.
3. To decide which targets should be allowed to leak through whenever there is insufficient time to effectively prosecute all targets.

The first task is called *target sequencing*; the second, *dwell time allocation*; and the third, *target rejection*. Collectively, all three tasks constitute a *target scheduling task*, to which we shall usually refer as the Target Scheduling Problem (TSP). The major goal of our program was to develop, test, and deliver real-time algorithms to perform this decision task for boost phase and midcourse threats ranging from a few to thousands of objects.

There has been a long-held belief in some battle management circles that this comprehensive decision task should be performed at the overall battle management level on a separate platform. In our opinion, the centralized control of complex, dynamic

networks whose nodes strongly depend on local data and conditions is usually doomed to failure due to excessive communication requirements, and because severe vulnerabilities are usually inevitable with a fully centralized approach. Furthermore, the memories and processing units of battle managers are usually fully occupied with overall battle decisions such as the allocation of *clusters* of targets to individual platforms. A distributed battle management policy is then far better because it lets individual platforms decide how to handle each element of their assigned cluster.

Conceptually, TSP can be classified as a "single-maching job shop scheduling problem".* While such problems have received considerable attention for many years, TSP is peculiar in so many complicating ways that the direct application of existing scheduling methods was not possible. Because the performance and robustness of real-time decisions algorithms is essentially determined by the quality of the physical models and data from which they are derived, most of the numerical and computational challenges experienced during our research resulted from our choice of modeling detail. Many of these difficulties could have been avoided by choosing a more convenient and necessarily more abstract modeling viewpoint, as others have done before us. But the need to achieve or at least to approach true operational optimality was a strong motivation to seek models that adequately reflect the physical reality within which engagements take place. In contrast to general and theoretical scheduling algorithms, *target* scheduling algorithms derive their decisions from a wide variety of real-time dynamic and stochastic data, and they must therefore be strongly physics-based.

A simple input-output description of the target scheduling decision function is provided in Fig. 1. In response to threat information updates, and given current informa-

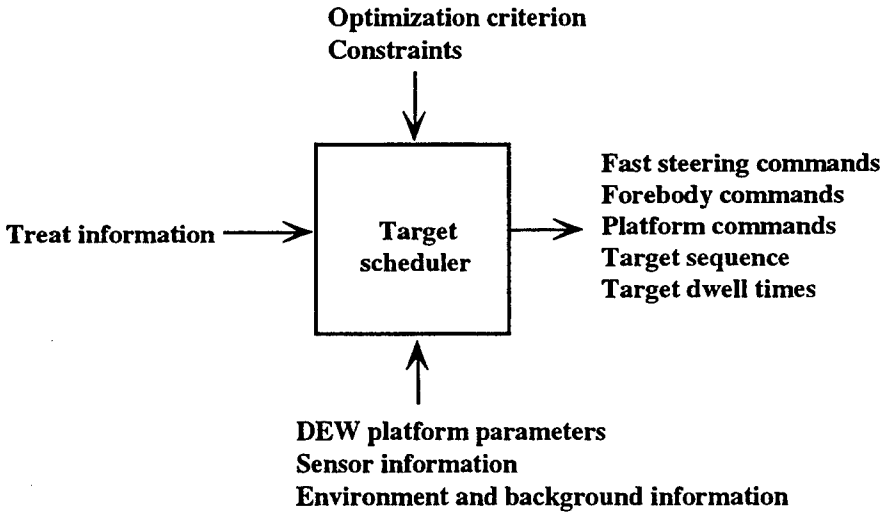

Figure 1. Input-output diagram of the target scheduling function.

* French (1982); Ibaraki and Katoh (1988); Baker (1974); Conway et al. (1967).

tion about the DEW platform, the external sensor network and the environment, the scheduling algorithm generates commands to the platform and to its control hardware. Output commands specify the start time and duration of DEW pulses, the desired pointing direction of the platform, the forebody, and the fast beam steering system. Any errors in the command-generating process can significantly reduce platform effectiveness, thereby severely increasing target leakage, so that the search for an optimal command sequence is essential. Any discontinuities in the command sequence can produce unwanted oscillations or vibrations in the platform structure, so special attention must be directed to produce *smooth* command sequences.

Approach

In our search for an optimal command sequence, we attempt to minimize the expected leakage cost, usually referred to as the (expected) *Leakage Risk*. Most of the difficulties in estimating the minimum risk are due to errors in evaluating the *actual* leakage risk in real time, and this imposes minimal modeling fidelity requirements. For instance, threat descriptions provided to the scheduling algorithms must include threat geometry, threat dynamics, and uncertainties in state and in parameters. DEW platform models must account for errors in Acquisition, Tracking, and Pointing (ATP) and errors in information handovers from external battle management and sensor platforms.

The risk minimization process is also subject to rigid energy and dynamics constraints. Platform hardware, for instance, is subject to hard limits in angular displacement, slew rates, acceleration, and mechanical damping. These limits increase platform response time, stretch out the operational timeline, and may require ignoring or *rejecting* some target so that others can be successfully prosecuted. Other constraints are due to restrictions in the opportunity windows within which threats must be processed. Each target has an opportunity window that opens at the target's availability, or *release time*, and closes at its *deadline*, the time by which the processing of the target must be completed, lest it leak through. Missed deadlines result in the sure leakage of one or more targets and, for anything but the sparsest threats, deadlines contitute our most severe constraints. Because their satisfaction was our highest scheduling priority, we have appropriately called our algorithm the Deadline-Driven Target Scheduling (DDTS) algorithm.

Target scheduling is easily the most computation-intensive decision process onboard a platform. Analytically intractable, the problem can only be solved in real time by using heuristics designed to approach optimum performance within acceptable errors. Reasonable upper bounds for such errors have not yet been found, even for most simplified versions of the target scheduling problem. To attain a satisfactory degree of confidence in the algorithm's performance, our approach involved a balanced mixture of theoretical and empirical arguments. On the theoretical side, analytical intractabilty was controlled by decomposing or reducing the problem into a few special cases whose theoretical structure is well understood. Examples of such reductions are

the well-known Traveling Salesman Problem, and the standard One-Machine Job Shop Scheduling Problem (French 1982). A significant level of confidence was achieved by verifying that our algorithm performed as expected on such well-known special cases.

On the empirical side, we developed a simulation package called THREATSIM to drive the DDTS algorithm with a wide variety of threat scenarios to fill the performance verification gaps left open by our theoretical approach. Designed to reveal the worst possible behavior of DDTS, THREATSIM is allowed to violate the laws of nature in order to produce synthetic threats that, although they will not be encountered in practice, may be used to "stress" the algorithm so that an adequate sensitivity assessment can be made.

While our scheduling algorithm automates tasks conventionally performed by humans, we have not recognized any benefits in applying concepts in Artificial Intelligence, Neural Nets, or Fuzzy Logic; therefore, we used only standard operation research and computer science methods.

Program Contributions

Contributions were made at two major levels during the three-year research period. At the conceptual level, we made significant improvements to scheduling technology by extending some conventional scheduling and network flow techniques to include stochastic and dynamic nodes and arcs. We thoroughly evaluated the DDTS algorithm, which is admittedly heuristic to some extent, by applying combinatorial complexity analysis techniques to important special cases. We also developed the THREATSIM algorithm to simulate a wide variety of threats and to reveal the behavior of DDTS in situations not encompassed by theoretical methods. This evaluation process confirmed that DDTS performs very well and is robust to input and parameter perturbations. Using appropriate trajectory and scheduling predictions, the algorithm also produces smooth command streams, thereby significantly reducing platform component "jerking".

At the applications level, we have developed, tested, and delivered to RL an efficient real-time target scheduling software package that optimally allocates the energy of a DE weapon by minimizing leakage risk during the boost phase and discrimination risk during the midcourse discrimination phase. With minor modifications, the DDTS algorithm can also be applied to many other resource allocation problems that may arise not only in strategic but also in tactical situations. Because the algorithm is strongly physics-based, it is also a sensitivity analysis tool with which the effects of parameter or input variations on the overall platform performance can be assessed. And with simulation drivers such as THREATSIM, DDTS may also be used as an end-to-end platform simulation structure.

1. Introduction

1.1. Background

The real-time allocation of weapon resources is operationally the most critical and computationally the most challenging battle management function on-board a weapons platform. On Directed Energy Weapon (DEW) platforms, this function requires the completion of three major tasks:

1. **Target Sequencing** to determine the order in which targets must be prosecuted.
2. **Dwell Time Allocation** to decide how long the DEW should dwell on any target.
3. **Target Rejection** to select the targets that should be allowed to leak through.

Collectively, these tasks constitute the Target Scheduling Problem (TSP). The purpose of this report is to present a solution to TSP for platforms designed to engage hundreds of targets during the boost-phase kill mission and thousands of objects during the midcourse discrimination mission (Fig. 1.1).

As a special case of the general Weapon-Target Assignment (WTA) problem,* TSP has received much attention in the applied literature, where it is usually expressed as a statistical decision problem and is thus solved as an optimization problem: This is also our approach, and we have chosen Expected Risk as our objective function for designing the target scheduler. In the boost phase, this is equivalent to minimizing the expected total value of the targets that leak through (*Expected Leakage Risk*), and in the midcourse discrimination phase it is roughly equivalent to minimizing the expected total value of the targets that are misidentified (*Expected Misidentification Risk*). When target values are all equal, these performance measures obviously reduce to the expected total *quantity* of targets that leak through or are misidentified, respectively.

In spite of the valued work of others, one or more of the following target scheduling difficulties have been insufficiently addressed in previous studies:

1. Deadline and Release Time (opportunity window) constraints.
2. Random and time-varying variables and parameters.
3. Uncertainties in target type and identification (ID).

* See Hosein and Athans (1990), Castañon et al. (1989a,b), Mealy and Megaloudis (1989).

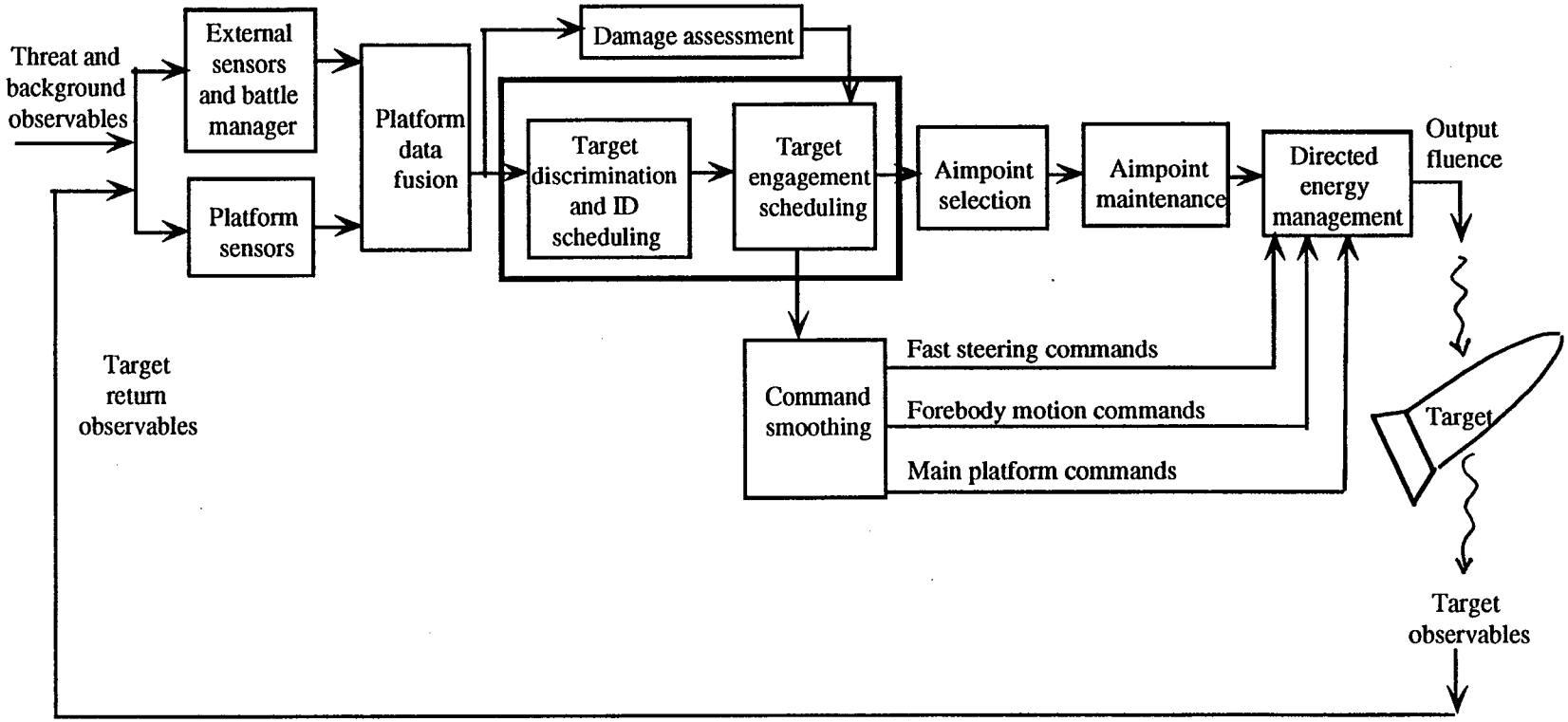

Figure 1.1. Target scheduling is the principal battle management function on a DEW platform.

4. Dynamic predictions of future threat and platform states are required to achieve optimal energy and time allocation.

5. Target values are unequal and may be uncertain.

6. Threats may not be aggregated and must be processed as point-by-point allocations.

7. When timelines are too tight to satisfactorily process all targets, some may have to be ignored or rejected.

8. Bayesian priors on target types and on other essential random variables must be fused with observations to obtain best posterior estimates.

9. Background and environmental noise processes must be considered.

10. Scheduling "horizons" must be defined to avoid scheduling targets too far into the future, where state predictions are unreliable.

11. Dynamic insertion and deletion of targets on the scheduling "stack" must be allowed.

Previous battle management discussions have also been too abstract for direct real-time application; therefore, proposed algorithms must be more physics-based. In sequencing targets to meet deadlines, for instance, assuming a linear relationship between target completion (retarget plus dwell) times and their angular separations may lead to unacceptable errors. The settling times of the platform hardware must be considered, and various non-linear constraints on hardware positioning, slewing, and acceleration must be obeyed. Furthermore, target dwell times strongly depend upon the lethality of the DEW, the vulnerability of the targets, and the accuracy and stability of the beam control subsystem. Decision algorithms must therefore (1) include reasonable beam-target interaction models and (2) account for various Acquisition, Tracking, and Pointing (ATP) errors. All this may strongly influence the design or selection of the optimization algorithms, as it did on the program discussed in this report.

At the theoretical level, similar shortcomings exist in the current literature, although two important special cases of our scheduling problem have received considerable attention. The first is the well-known Traveling Salesman Problem,* which applies to cases without time windows and where dwell times and retarget times are fixed. But

* Lawler (1971); Lawler et al. (1985); Norback and Love (1977); Flood (1956); Garey and Johnson (1979); Eddy (1977); Sedgewick (1988).

even then a heuristic approach is needed (Laporte 1992) because the classical traveling salesman problem is itself Nondeterministic Polynomial (NP)-hard (Garey and Johnson 1979). The second special case is known as the "One-Machine Job Scheduling Problem with Ready and Due Times"; this was solved in Kise et al. (1978) using the quantity of late ("tardy") jobs as an objective function, and with some mild ordering assumptions on the ready (release) times and deadlines. In the Kise study, no waiting or "transition" times were allowed, all jobs had equal value, and all problem parameters were constants.

In spite of recent formal efforts to mix both problems (Daniels 1990, Tsitsiklis 1992), we had no choice but to include several heuristic arguments in our scheduling approach. But heuristic arguments were used only to *extend* the well-known solutions of important special cases to our new problem, thereby providing considerable error control since our algorithms were designed to agree with others on those special cases. We completed a thorough review of other potentially useful methods to accomplish these extensions, including Artificial Intelligence, Neural Nets, and Fuzzy Logic, but we have not recognized any benefits in using such methods. Even though our algorithms are easily parallelizable, we have intentionally ignored hardware implementation issues.

1.2. The Boost Phase

During this phase, the platform's primary mission is to destroy boosters to minimize the expected leakage risk, subject to opportunity window constraints. At regular intervals, the target scheduler receives threat updates from external platforms and from on-board sensors and monitors. For each interval, the scheduler must do the best it can with the information acquired during that interval; each interval is treated as a new phase in the scheduling process, even though information may remain valuable for several update intervals.

1.3 The Midcourse Discrimination Phase

During midcourse, target Re-entry Vehicles (RVs) will typically be surrounded by several decoys and replicas, thereby forming "clouds" of objects whose type must be resolved prior to allocating energy or dispatching interceptors. This *discrimination* function also involves a scheduling task much like that in the boost phase. In this task, a subtle tradeoff must be resolved between two risks, a *classification risk* and a *rejection risk* (Fig. 1.2).

The classification risk is the total expected value of the target lost due to improper identification, and it is computed using conventional Bayesian classification methods. The Bayesian optimization approach is not bound by the strong and often impractical assumptions required to apply the notion of "k-factor", an approach still widely used in some sensor technologies (Holmes and Rocklin 1990; Rocklin and Tolleson 1986).

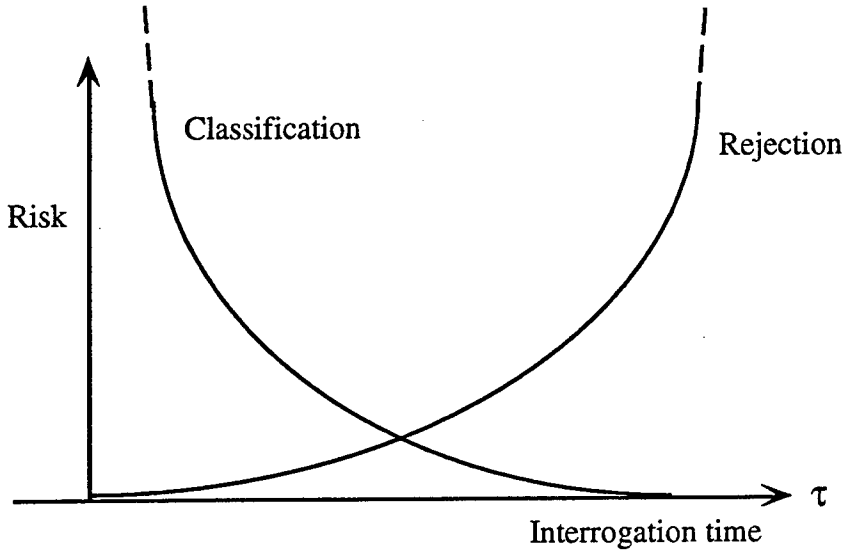

Figure 1.2. As interrogation time by the DEW increases, the classification risk decreases and the risk of target rejection increases.

For each misidentified decoy or replica, we assume that only the cost of one interceptor or lethal energy dose is wasted, depending upon the midcourse interception mode assumed. We also assume that each misidentified RV leaks through. The rejection risk is the total expected value of the targets that are not processed by their due date.

As target interrogation (dwell) time is increased, classification risk obviously decreases and rejection risk increases because the time line is stretched out. The sum of both risks can thus be minimized by finding an optimal value τ^* of the interrogation time.

Inputs to the platform include the dynamic state of cloud *centroids*, but not of individual cloud elements (RVs, decoys, replicas). The platform is thus expected to resolve each cloud into its contituent elements, but we have not included that function in our scheduling algorithm.

Target interrogation is accomplished with a Neutral Partricle Beam (NPB), and target classification algorithms must include suitable beam-target interaction models and a neutral particle collection model for each of the deployed sensors. Although data fusion is an issue because we have a sensor *network*, we assume the simple fusion policy where the network output is equal to the particle count of the sensor whose signal-to-noise ratio is greatest.

1.4 Report Structure

Organizing this report was a very difficult task: our discussion spans many levels of discourse and there is considerable technical overlap among many of the issues addressed in this program. We also lost track to some extent of exactly where the greatest amount of resources were expended throughout this three-year research program. In retrospect, it seemed that technical difficulties were so interdependent that, as soon as we thought a problem was laid to rest, it was resuscitated by new issues deferred earlier. This is a familiar situation when maximal efficiency is demanded from an algorithm, so we settled on the following organization.

In Chapter 2 we lay the decision-theoretic foundations for our scheduling algorithm, and we carefully define the stochastic processes that describe the various measurement processes that drive the decision problem. We state what we mean by a "decision maker", and, after defining problem constraints, we present a precise discussion of the optimization issue. We then summarize our approach to solving the optimization problems for the boost and midcourse phases.

In Chapter 3 we focus on the boost phase, and we go deeper into the structure of the scheduling algorithms. Drawing from the traveling-salesman literature, we show how optimal tours are constructed and improved as needed, how targets are rejected when necessary, and how optimal dwell times are derived. In Chapter 4, the boost phase Deadline-Driven Target Scheduling (DDTS) algorithm is translated into software. We describe in detail how the running time of DDTS is minimized by employing efficient sorting and searching techniques, and we also discuss the tour construction software in detail. We conclude by including an important extension from space-based platforms to ground-based DEWs.

We address the midcourse discrimination phase in Chapter 5. Since the basic scheduling issue is similar to that in the boost phase, we concentrate on the inner optimization loop concerned with Target Classification, which we address using Bayesian classifiers, as we discussed earlier. We especially emphasize the physical processes assocated with NPB target interrrogation, and we carefully account for the various processes that contribute to the total particle count at each NPB sensor. We conclude Chapter 5 with a detailed account of how *cloud* dwell times and retarget times are calculated for two major cloud distributions consisting of RVs, replicas, and decoys.

In Chapter 6 we report our testing work with DDTS. We start with a description of THREATSIM, a threat simulator designed to drive DDTS as an end-to-end platform simulator. We briefly interpret each of eight major tests conducted with the algorithm. We close the report with Chapter 7, where we present some conclusions from this extended research program and outline some promising directions for future research.

2. Target Scheduling: A Statistical Decision Problem

The principal purpose of a target scheduling algorithm is to decide, within specific time and resource constraints, in what order a collection of dynamic and probabilistic targets should be processed and how many resources should be allocated to each target so that the risk of target leakage is minimized. In such general terms, the Target Scheduling Problem (TSP) appears as just another statistical decision problem. As we suggested in Chapter 1, however, a closer inspection reveals a problem fraught with complications that have persistently resisted the direct application of existing decision-theoretic and optimization methods. These difficulties are sufficiently subtle and pernicious to render essential a careful definition of the TSP before they can be understood. A clear articulation of the issues is also essential for the recognition of familiar problems and the application of past and current work of others. We also found a decision-theoretic formulation of the TSP very useful in obtaining an effective model that reveals the real-time computational challenges lurking in the background.

In this chapter we do the following. We start by stating what we mean by a "statistical decision problem", and we carefully describe the canonical elements of such a problem. Then we define the TSP by providing the required physical and geometric interpretation for the canonical elements. We originally intended to provide a discussion of the decision problem that does not distinguish between the Boost Phase mission and the Midcourse mission in order to concentrate on principles and methodology, but such a generic treatment would have been too abstract. There are important conceptual and operational differences between the two missions, so we chose instead to emphasize the boost phase in this chapter, although several excursions into the Midcourse problem are made whenever conceptual differences prevented a boost phase interpretation.

To adequately capture the dynamics and statistics of the TSP and to reflect our computer implementation approach, we use the general framework of stochastic processes (Breiman 1968, Chung 1974) to describe the decision problem and all its ingredients.

2.1 Statistical Decision Theory: Some Defintions

Consider a stochastic observation process $\mathcal{X} = (X_1, \ldots, X_i, \ldots, X_n)$ consisting of n random observation variables $X_i, i = 1, \ldots, n$, whose underlying *probability space* (Breiman 1968, Chung 1974) is $P_\Omega =< \Omega, \mathcal{B}, P >$. Each X_i is a $\mathcal{B}$-measurable function from Ω to R which produces an observation $X_i(\omega) = x$ when the true *state of nature is* ω. Whenever an *event* $A \in \mathcal{B}$ occurs, an event $X_i(A)$ is observed by X_i with probability $P(A)$.

Next consider an *action space* $\mathcal{A}$ and a *decision maker* $DM =< d, L >$ who chooses an action $d(x) \in \mathcal{A}$ whenever he observes $X_i(\omega) = x \in R$, and incurs a loss $L(\omega, d(x))$ whenever he completes that action. *A rational* criterion (Berger 1980) for choosing that course of action is the *expected* risk

$$\mathcal{R}(d) = E_P(L(\omega, d(x))), \tag{2.1}$$

the expected value of the loss function $L(\omega, d(x))$ with respect to the probability measure P and the decision function $d : R \rightarrow \mathcal{A}$. In dynamical problems such as the TSP, a *realization viewpoint* (Breiman 1968) is often useful where the index set $I = \{1, \ldots, i, \ldots, n\}$ is a discrete time scale and, for each $\omega \in \Omega$, the sequence of observations $(X_1(\omega), \ldots X_i(\omega), \ldots X_n(\omega))$ is considered a function of time i, with ω held fixed. This function, denoted by $\mathcal{X}(\omega)$, is a *realization* of the stochastic process $\mathcal{X}$. Whenever a decision maker chooses a course of action on the basis of the realizations $\mathcal{X}(\omega)$ of $\mathcal{X}$, he is called a *sequential decision maker*, and the decision rule d is a function d: $\mathcal{X}^R \rightarrow \mathcal{A}$ from the set $\mathcal{X}^R$ of realizations of $\mathcal{X}$ into the action space $\mathcal{A}$, taking each $x^n \in \mathcal{X}^R$ into an action $d(x^n) \in \mathcal{A}$. This situation is captured by the following:

Definition

An *instance of a statistical decision problem** is a fourtuple

$$ISDP =< \mathcal{X}, DM, F, \mathcal{R} > \tag{2.2}$$

where:

- $\mathcal{X}$ is a stochastic observation process on a probability space $P_\Omega =< \Omega, \mathcal{B}, P >$, with *realizations* $\mathcal{X}^R$.
- DM a decision maker with *decision function* $d : \mathcal{X}^R \rightarrow \mathcal{A}$ and *loss function* $L : \Omega \times \mathcal{A} \rightarrow \mathcal{L}$, the *loss space.*
- F is the set of *feasible* decision functions (Papadimitriou and Steiglitz 1982).
- $\mathcal{R}$ is a *risk function* (objective function) which maps the loss function L, observation process $\mathcal{X}$, decision function d, and probability space P_Ω into a *decision risk* $\mathcal{R}(L, d, \mathcal{X}, P_\Omega)$.

The basic objective in a statistical decision problem is to choose a feasible course of action $d \in F$ which minimizes the risk $\mathcal{R}$. In this report, we confine our attention to situations where $\mathcal{R}$ is the expected risk as expressed in Eq. (2.1).

Given a collection of problem instances, one for each $d \in F$, a *statistical decision* problem is an optimization problem:

---

* Papadimitriou and Steiglitz (1982); Ferguson (1967); Berger (1980).

Definition

Consider a collection of feasible instances $\{ISDP_d : d \in F\}$. The *statistical decision problem* is to find a decision rule $d^* \in F$ such that

$$\mathcal{R}(L, d^*, \mathcal{X}, P_\Omega) = \min_{d \in F} \{\mathcal{R}(L, d, \mathcal{X}, P_\Omega)\} = \mathcal{R}^* \,. \tag{2.3}$$

This is the *minimum risk*, and d^* is called a *solution* to the decision problem.

When no specific choice of $\mathcal{X}$, DM, F, or $\mathcal{R}$ is intended, and to simplify notation, we shall occasionally refer to the fourtuple $< \mathcal{X}, DM, F, \mathcal{R} >$ as *the decision problem*, with the understanding that the optimization process described in Eq. (2.3) is implicitly included.

2.2 A Definition of the Target Scheduling Problem (TSP)

The abstract definitions in the previous section were useful in providing a conceptual interpretation of the TSP as a canonical decision problem. But the causes and consequences of scheduling decisions can be properly understood only if an appropriate physical and geometric interpretation is assigned to each element of the canonical problem. Such an interpretation is developed in this section. First, we define the elements of the TSP; then we state the optimization problem.

2.2.1. Principal Elements of the TSP

Recall the definition of a statistical *decision problem instantiation* (Eq. (2.2)):

$$ISDP =< \mathcal{X}, DM, F, \mathcal{R} > \tag{2.4}$$

In the TSP, the decision maker observes the stochastic states of the targets, the sensors, the DEW, and the environment via a process $\mathcal{X}$ whose underlying probability space P_Ω describes all the uncertainties that corrupt this observation process.

The *"decision maker"* $DM =< d, L >$ has a *sequential decision rule* (Berger 1980) which generates various decisions and commands from the observation process $\mathcal{X}$. The rule specifies the order in which targets are prosecuted and the time and energy allocated to each target, and then selects those targets which must be ignored. The rule also produces position commands to the system controlling the platform main body, the forebody, and the fast steering subsystem. The resulting actions incur a loss specified by the loss function L, which accounts for the different values of individual targets that leak through because of missed deadlines, ATP errors, or insufficient DEW lethality.

The constraint or feasibility set F is determined by a constraint on d which prohibits (1) scheduling targets before their release time or after their deadline and (2) exceeding resource limits. Constraints are implicitly defined by deadlines, release times, energy limits, a *target completion time* function, and an *energy consumption* function.

In this report, risk is defined as the expected value (E) of a loss function (L) with respect to the probability of leakage P_L of each target, usually formalized as a risk criterion $H =< L, P_L, E >$. For both the boost and the midcourse phase, explicit forms for the expected risk will be derived in the appropriate sections.

This interpretation, together with the earlier canonical definition, motivate the following representation.

Definition

The Target Scheduling Problem (TSP) is a statistical decision problem

$$TSP =< \mathcal{X}, DM, F, H > \tag{2.5}$$

whose elements are given in the following subsections of Section 2.2.1.

2.2.1.1. The Observation Process $\mathcal{X}$. Four physical processes are observed by the decision maker to arrive at scheduling decisions: the targets, the sensors, the environment, and the DEW platform itself. The decision maker's measurement process is thus a function of four subprocesses:

$$\mathcal{X} = \mathcal{X}\,(\mathcal{X}_T,\, \mathcal{X}_D,\, \mathcal{X}_E,\, \mathcal{X}_S) \tag{2.6}$$

where
$\mathcal{X}_T$ is the target process.
$\mathcal{X}_D$ is the weapons platform process.
$\mathcal{X}_E$ is the environment process.
$\mathcal{X}_S$ is the sensor process.

As we shall see below, these processes are characterized by *states* and *parameters*. We assume that the decision maker must infer all parametric information from state measurements or from prior knowledge.

2.2.1.1.1. The Target Process $\mathcal{X}_T$. Consider a set of targets $\mathcal{S} = \{T_i : i \in I\}$, where I is some (ordered) index set of size (cardinality) $\mid I \mid$. The ordering on I induces an ordering on $\mathcal{S}$, producing an ordered *target set* $\mathcal{T} =< T_i : i \in I >$ of $\mid I \mid$ targets, also called a *target sequence*. Whenever we refer to a "target set" in this report, we shall tacitly assume that this set is ordered by some index set I, or a permutation thereof.

The *target process* is a collection $\mathcal{X}_T = \{\mathcal{X}_i : i = 1, \ldots, \mid I \mid\}$ of individual target processes $\mathcal{X}_i$, one for each $T_i \in \mathcal{T}$.

Each target process is itself a random process defined by a random parameter vector and a random state vector, as follows:

Definition

$$\mathcal{X}_i = < \pi_i^T(Q_i^T(t,\omega^T),t,\omega^T), h^T(Q_i^T(t,\omega^T)), P_\Omega^T > \tag{2.7}$$

where:

t	is the time variable.
P_Ω^T	is a probability space $< \Omega^T, \mathcal{B}^T, P^T >$.
$\omega^T \in \Omega^T$	is a sample value from the sample space Ω^T.
$\mathcal{B}^T$	is a σ-algebra of events from Ω^T.
P^T	is a probability measure on $\mathcal{B}^T$.
$Q_i^T(t,\omega^T)$	is the *state* of T_i at time t when the *state of nature* is ω^T.
$\pi_i^T(Q_i^T(t,\omega^T),t,\omega^T)$	is the value at time t of the *parameter vector* of T_i when the state is $Q_i^T(t,\omega^T)$ and the state of nature is ω^T.
$h^T(Q_i^T(t,\omega^T))$	is a function representing the observation of the state $(Q_i^T(t,\omega^T))$.

Informally, each target T_i is a probabilistic object whose underlying uncertainties are captured by P_Ω^T, whose description is parametrized by π_i^T, and whose dynamic state Q_i^T is observed through sensors h. The parameter vector $\pi_i^T(\cdot)$ typically contains parametric information about the target's hardness, vulnerability, and type.

The dynamic state of a target T_i is predicted Δt seconds into the future from some initial handover time t_0 (initialization time) by assuming that T_i travels along a straight line or along a constant-radius circle centered at the earth. We thus assume a simple first-order dynamical model where the future position $X_i^T(t_0 + \Delta t)$ in earth-centered coordinates (indicated by subscript E) is extrapolated from the initial position $(x_{E,0},\ y_{E,0},\ z_{E,0})$ and velocity $(\dot{x}_{E,0}, \dot{y}_{E,0}, \dot{z}_{E,0})$ as follows (superscript T omitted):

If the angular position of the target in the earth-centered spherical coordinate system of Fig. 2.1 is $(\theta_E(t_o),\ \phi_E(t_0))$ at time t_0, then at time $t = t_0 + \Delta t$,

$$\theta_E(t) = \theta_E(t_0) + \dot{\theta}_E(t_0)\Delta t$$

and

$$\phi_E(t) = \phi_E(t_0) + \dot{\phi}_E(t_0)\Delta t \tag{2.8}$$

where

$$\theta_E(to) = tan^{-1}\left(\frac{y_{E,0}}{x_{E,0}}\right),$$

$$\phi_E(t_0) = \cos^{-1}\left(\frac{z_{E,0}}{\left((x_{E,0})^2 + (y_{E,0})^2 + (z_{E,0})^2\right)^{1/2}}\right),$$

$$\dot{\theta}_E(t_0) = \frac{x_{E,0}\dot{y}_{E,0} - y_{E,0}\dot{x}_{E,0}}{(x_{E,0})^2 + (y_{E,0})^2}\,, \text{and}$$

$$\dot{\phi}_E(t_0) = \left(\frac{1}{((x_{E,0})^2 + (y_{E,0})^2)^{1/2}}\right)\left(\frac{z_{E,0}(x_{E,0}\dot{x}_{E,0} + y_{E,0}\dot{y}_{E,0} + z_{E,0}\dot{z}_{E,0})}{(x_{E,0})^2 + (y_{E,0})^2 + (z_{E,0})^2} - \dot{z}_{E,0}\right).$$

The derivation of Eq. (2.8) is straightforward, and it can be found in Appendix A.

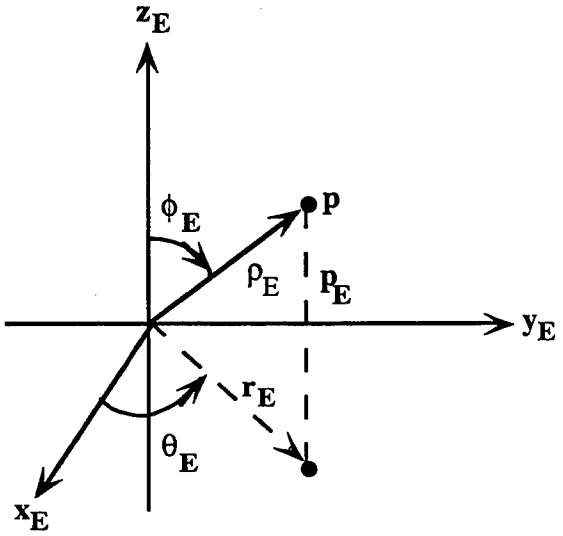

Figure 2.1. Transformation to spherical coordinates.

In rectangular earth-centered coordinates, the predicted target position at $t = t_0 + \Delta T$ is then $X_E(t) = \Big(x_E(t), y_E(t), z_E(t)\Big)$, where

$$\begin{aligned} x_E(t_0 + \Delta T) &= x_E(t) = \rho_E \cos\theta_E(t) \sin\phi_E(t)\,, \\ y_E(t_0 + \Delta T) &= y_E(t) = \rho_E \sin\theta_E(t) \sin\phi_E(t)\,, \\ z_E(t_0 + \Delta T) &= z_E(t) = \rho_E \cos\phi_E(t)\,, \end{aligned} \tag{2.9}$$

and where

$$\rho_E = \Big((x_E)^2 + (y_E)^2 + (z_E)^2\Big)^{1/2}.$$

Expressing this new position in platform coordinates indicated by subscript P,

$$X_P(t) = R(X_E(t) - D_E(t)) \tag{2.10}$$

where $D_E(t)$ is the new platform position at $t = t_0 + \Delta t$, and R is the rotation matrix relating the platform coordinates to the earth-centered coordinate system.

2.2.1.1.2. The DEW Platform Process $\mathcal{X}_D$. Similar to targets T_i, the DEW platform is a stochastic process modeled by the triple

$$\mathcal{X}_{\mathcal{D}} =< \pi^D(Q^D(t,\omega^D),t,\omega^D), h^D(Q^D(t,\omega^D)), P_\Omega^D) > \tag{2.11}$$

where the interpretation of π^D, h^D, and P_Ω^D is analogous to that for each T_i.

The future platform position $X^D(t_0+\Delta t)$ Δt seconds into the future is estimated in earth coordinates, as for targets T_i (subscript E omitted and ω held fixed):

$$X^D(t_0+\Delta t) = (x^D(t_0+\Delta t), y^D(t_0+\Delta t), z^D(t_0+\Delta t))$$

where

$$\begin{aligned}
x^D(t_0+\Delta t) &= \rho^D \cos\theta^D(t_0+\Delta t)\sin\phi^D(t_0+\Delta t)\,. \\
y^D(t_0+\Delta t) &= \rho^D \sin\theta^D(t_0+\Delta t)\sin\phi^D(t_0+\Delta t)\,. \\
z^D(t_0+\Delta t) &= \rho^D \cos\phi^D(t_0+\Delta t)\,. \\
\theta^D(t_0+\Delta t) &= \theta^D(t_0) + \dot{\theta}^D(t_0)\Delta t\,. \\
\phi^D(t_0+\Delta t) &= \phi^D(t_0) + \dot{\phi}^D(t_0)\Delta t
\end{aligned} \tag{2.12}.$$

As discussed in later chapters, the parameter vector $\pi^D(\cdot)$ of a platform typically contains a parametrized description of the platform power, bias, jitter, beamwidth, and DEW wavelength.

2.2.1.1.3. The Environment Process $\mathcal{X}_E$. The physical environment for an engagement is described by an underlying probability space and a random parameter vector, as follows:

$$\mathcal{X}_E =< \pi_E(Q_E(t,\omega_E)), h^E(Q^E(t,\omega^E)), P_\Omega^E >, \tag{2.13}$$

where

P_Ω^E $= (\Omega^E, \mathcal{B}^E, P_E)$,

$Q^E(t,\omega_E)$ is the *state of the environment* at time t, when the state of nature is ω_E, and

h^E represents the measurement of the environmental state.

The "environment" typically specifies ambient conditions which influence the outcome of engagements, but which cannot be directly attributed to targets or platforms. "Environment" typically includes descriptions of ambient noise, light levels, and background.

As a good example relevant to midcourse discrimination using particle beams, consider the random particle background count observed by some sensor in space. Then

P_Ω^E could represent the particle generation process and π_E could be the expected particle count at some specific location in space, at the sensor for instance. (See Chapter 5 for a more specific discussion).

2.2.1.1.4. The Sensor Process $\mathcal{X}_S$. Similar to the target process $\mathcal{X}_T$, $\mathcal{X}_S$ is a collection $\{S_j : j = 1, \ldots, |J|\}$ of sensor processes S_j, each described by a random parameter vector and a random state vector, as follows:

$$S_j =< \pi_j^S(Q_j^S(t,\omega^S),t,\omega^S),\ h^s(Q_j^S(t,\omega^S),\ P_\Omega^S) > , \tag{2.14}$$

where:

t is time.

$Q_j^S(t,\omega^S)$ is the *state* of S_j at time t when the state of nature is ω^S.

$\pi_j^S(Q_j^S(t,\omega^S),t,\omega^S)$ is the value at time t of the parameter vector of S_j when the state of S_j is $Q_j^S(t,\omega^S)$ and the state of nature is ω^S.

$h^S(Q_j^S(t,\omega^S))$ is a function representing the observation of the state $Q_j^S(t,\omega^S)$, typically its expected value at time t.

$P_\Omega^S =< \Omega^S, \mathcal{B}^S, P^S >$ is the probability space underlying the parameter vector and state of S_j.

In contrast to targets and platforms, which may be viewed as point masses since their orientation is not a critical issue in our analysis (except for their aspect angle θ), sensors require a higher-dimensional representation. The major reason is that sensors are typically directional, and they also experience precessions, rotations, and tumbles as they move through space. We use 12 degrees of freedom to represent the state of sensors (see Fig. 2.2):

$$Q^S =< X_E^S,\ \dot{X}_E^S;\ \alpha^S,\ \beta^S,\ \gamma^S;\ \dot{\alpha}^S,\ \dot{\beta}^S,\ \dot{\gamma}^S > , \tag{2.15}$$

where:

X_E^S is the sensor position in earth coordinates.

$\dot{X}_E^S$ is the sensor velocity in earth coordinates.

α_S^S and $\dot{\alpha}_S^S$ are the *rotation* and the *rotation rate* of the sensor about the z^S-axis, respectively.

β_S^S and $\dot{\beta}_S^S$ are the rotation and rotation rate about y^S, respectively.

γ_S^S and $\dot{\gamma}_S^S$ are the rotation and rotation rate about x^S, respectively.

The angular position state of the sensor is thus $(\alpha_S^S,\ \beta_S^S,\ \gamma_S^S)$ and the *rotation rate* is $(\dot{\alpha}_S^S\ ,\ \dot{\beta}_S^S,\ \dot{\gamma}_S^S)$.

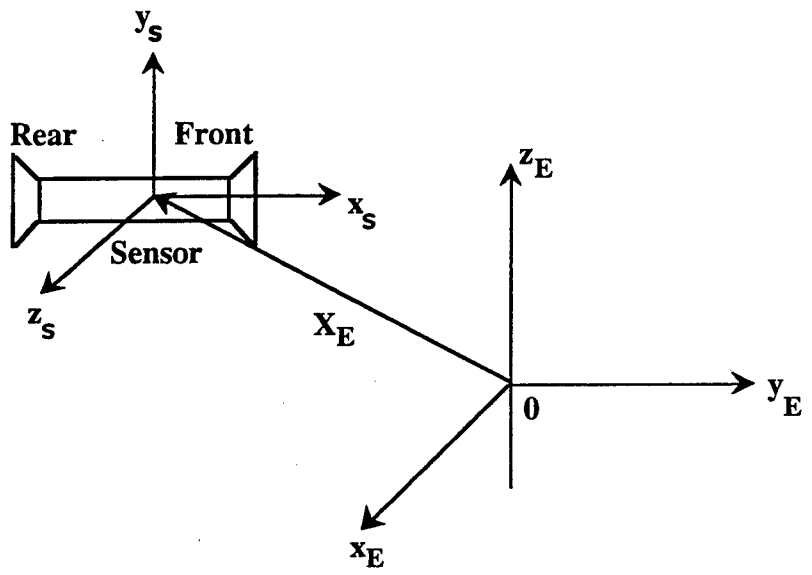

Figure 2.2. The sensor coordinate system (x_S, y_S, z_S) is a translation of the earth-centered coordinate system (x_E, y_E, z_E) by the displacement vector X_E.

When the state of a sensor Δt seconds into the future is needed, we proceed as before for the position state X_E^S of the sensor. For the rotation state, analogously,

$$\begin{aligned}
\alpha_S^S(t_0 + \Delta t) &= \alpha_S^S(t_0) + \Delta t\ \dot{\alpha}_S^S(t_0)\ . \\
\beta_S^S(t_0 + \Delta t) &= \beta_S^S(t_0) + \Delta t\ \dot{\beta}_S^S(t_0)\ . \\
\gamma_S^S(t_0 + \Delta t) &= \gamma_S^S(t_0) + \Delta t\ \dot{\gamma}_S^S(t_0)\ .
\end{aligned} \tag{2.16}$$

The sensor parameter π^S of Eq. (2.14) has four principal components:

$$\pi^S =< FOV, \eta, A_D, \tau_D >\ , \tag{2.17}$$

where

FOV is the sensor *field of view*, $FOV =< e_S, \theta_S, \phi_S >$.
e_S is the *sensing direction* (unit vector).
η is the detector conversion factor (electrons/particle).
ϕ_S is the sensor elevation angle.
θ_S is the *sensing angle* (spherical).
A_D is the *effective detector area.*
τ_D is the *detector response time.*

2.2.1.1.5. The Overall Measurement Process $\mathcal{X}$. The DEW platform collects information from on-board and external sensors at sequential times $\{t_k : k = 1, 2, \ldots\}$ determined by a given update rate. At any such time t_k, the decision maker

DM receives state summaries from trackers and sensors via the measurement functions h^T, h^S, h^D, and h^E. In this report, the first three functions produce mean and variance estimates of the positions and velocities of targets, sensors, and the DEW platform, and h^E generates the mean and variance of the background and environmental noise processes. The overall measurement process of Eq.(2.6) is thus simply a vector process

$$\mathcal{X}(\mathcal{X}_T, \mathcal{X}_D, \mathcal{X}_E, \mathcal{X}_S) = \Big[\{(\mu_i^T, \sigma_i^T) \ : i = 1, \ldots, |I|\},$$
$$(\mu^D, \sigma^D), (\mu^E, \sigma^E), \{(\mu_j^S, \sigma_j^S) \ : j = 1, \ldots, |J|\}\Big] \ , \tag{2.18}$$

where t_k was conveniently omitted, and the meaning of symbols is obvious.

2.2.1.2. The Decision Maker DM. In this report, a decision maker is a pair $DM =< d, L >$ where d is a *decision rule* that maps realizations $X(\omega)$ of the observation process $\mathcal{X}^R$ into actions $a \in \mathcal{A}$, and L is a *loss function* that assigns to each action a and state of nature $\omega \in \Omega$ a (random) cost or *loss* $L(\omega, a) = c \in \mathcal{L}$. Referring to the observation process $\mathcal{X}$ described in Section 2.2.1.1, we assume that the decision maker must infer all information about the parametric structure of targets, sensors, and the environment exclusively from state measurements or from prior knowledge.

2.2.1.2.1. The Decision Rule d. Each action taken by the decision maker specifies a target sequence, an allocation of dwell time and dwell energy, targets to be rejected or ignored, and other platform commands discussed earlier. For both the midcourse and the boost phase, the decision rule d is a complex algorithm that has no succinct mathematical representation. As we describe in later sections, target sequences are chosen by first ordering targets in accordance with their deadlines and release times, earliest deadlines first. If a sequence is not optimal, correction methods are applied until an optimal sequence is found or until a maximum computational time limit is reached.

During the boost phase, dwell times are obtained from leakage probabilities computed in an outer optimization loop, each probability providing a unique target dwell time.

When all deadlines cannot be met or under certain conditions where time lines must be shortened, some targets may have to be rejected or ignored. This is accomplished by applying a rejection criterion discussed in detail in Section 3.2.1.

As a matter of notation, *sequences* are viewed as permutations (reorderings) of some index set (Loomis and Sternberg 1968). Consider the set $\Pi = \{\pi : I \to I\}$ of all permutations of the index set I. A permutation $\pi \in \Pi$ reorders the index set I in such a way that index $\pi\ (i)$ is placed in position i, and $\pi^{-1}(j)$ is the index which was in the jth position prior to the permutation operation. More generally, consider a job or target sequence $\mathcal{T}_{(I)} =< T_1, \ldots, T_i, \ldots, T_{|I|} >$. Then the application of permutation π to I produces the reordered sequence

$$\mathcal{T} \cdot \pi(I) =< T_{\pi(1)}, \ldots, T_{\pi(i)}, \ldots, T_{\pi(|I|)} > \ , \tag{2.19}$$

which we denote by $\mathcal{T}_\pi$, and where

$T_{\pi(1)}$ is the target in the first position after permutation,
$T_{\pi(i)}$ is the target in the ith position after permutation,
$T_{\pi(|I|)}$ is the target in the $|I|$th position after permutation, and
$\mathcal{T}$ is the function assigning names or labels to the indices in $\pi(I)$.

Similarly, we denote by $T_{\pi^{-1}(j)}$ the target which *was* in the jth position prior to the permutation operation.

When some targets are rejected in accordance with some rejection criterion, only a *subsequence* of targets is prosecuted, and an order-preserving subsequence mapping $r : I \rightarrow I' \subset I$ is induced on I (Apostol 1974). This mapping takes any sequence into a subsequence $< s_{r(1)}, \ldots, S_{r(|I'|)} >$. The new target sequence is thus

$$\mathcal{T} \cdot \pi \cdot r(I) =< T_{\pi(r(1))}, \ldots, T_{\pi(r(|I'|))} > \ . \tag{2.20}$$

We shall denote subsequences of $\mathcal{T}_\pi$ by $\mathcal{T}'_\pi$ unless the rejection criterion must be explicitly included, in which case we denote them by $\mathcal{T}_{\pi \cdot r}$. We also denote a *set* of rejected targets by $\mathcal{T}_r$ and its index set by I_r.

With this notation, we can now succinctly describe each action of the decision maker as a triple

$$a =< \pi, t^D, I_r > , \tag{2.21}$$

where

π specifies the sequence chosen by DM,
t^D specifies the dwell times for all targets, and
I_r specifies the set of targets rejected by DM.

2.2.1.2.2. The Loss Function L. When the environment of an engagement is described by a "state of nature" $\omega \in \Omega$, each action $a \in \mathcal{A}$ incurs a random loss $L(\omega, a) \in \mathcal{L}$. Losses are thus random variables because they depend upon random engagement conditions.

But the losses also depend upon *when* they are incurred, and they are therefore also sequence-dependent. If a deadline is missed, for instance, a target may have to be ignored, so the earlier a target can be processed, usually the better. But if a target is scheduled *too* early, its release time may be violated, and additional losses may arise. Also, the probability of neutralizing a target depends on *when* the target is engaged. In some cases, waiting for a target may even *improve* weapon performance. Considering that the evaluation of leakage risk requires that losses be multiplied by their probability of occurrence on a target-by-target basis, respecting the order in which losses are incurred is essential, and losses must be expressed as *sequences* of losses.

As outlined earlier, each action $a \in \mathcal{A}$ and state of nature $\omega \in \Omega$ result in a set $\mathcal{T}_r = \{T_{\pi(i)} : i \in I_r\}$ of *rejected targets* and in a set $\mathcal{T}_m = \{T_{\pi(i)} : i \in I_m\}$ of *missed*

targets. Observe that this is *not* a probabilistic statement: the specific action and state of nature *together* cause $\mathcal{T}_r$ and $\mathcal{T}_m$ *for sure.* To find the *probabilities* that targets are either rejected or missed, we need to know the *distribution* of the states of nature, and these are discussed later when miss probabilities (p_M) are derived.

To define the sequential loss function L, we associate with each target $T_i \in \mathcal{T}$ a fixed *value*

$$V_i = V(T_i) = \alpha_i V^\circ(T_i) \in \mathcal{V} \tag{2.22}$$

where

$V^\circ(T_i)$ is the *intrinsic value* of T_i.
α_i is the *deadline hardness* of T_i.
$\mathcal{V}$ is the *value space.*

Accordingly, if I_r and I_m are the (ordered) index sets of $\mathcal{T}_r$ and $\mathcal{T}_m$, respectively, the loss function produces, for each state-action pair (ω, a), a *loss subsequence,*

$$L(\omega, a) =< V_{\pi(i)} : \ i \in I_r(\omega, a) \cup I_m(\omega, a) >, \tag{2.23}$$

indicating the order in which rejection losses (I_r) and miss losses (I_m) are incurred.

Expressed as a full sequence with $|I|$ terms,

$$L(\omega, a) =< V_{\pi(i)} \mathcal{X}^L_{\pi(i)}(\omega, a) : \ i = 1, \ldots, |I| > , \tag{2.24}$$

where

$$\mathcal{X}^L_{\pi(i)} = \left\{ \begin{array}{ll} 1, & i \in I_r(\omega, a) \cup I_m(\omega, a) \\ 0, & \text{otherwise} \end{array} \right\}$$

In Sections 2.2.1.4 and 2.2., this loss function will be combined with the probabilities on I_r, I_L, and $V_{\pi(i)}$ to obtain an expression for the overall risk.

2.2.1.3. The Feasible Set F. In complex problems such as the TSP there is no convenient closed-form definition of the feasible set F. Usually, a test function is defined, together with a *feasible region.* A solution is then *feasible* if the value of the test function for that solution falls in the feasibility region.

Two major constraints are considered in this report—time constraints and resource constraints—and their test functions are called the *target completion time function* (f_C) and the *energy consumption function* (f_E), respectively. Accordingly, we represent the *feasibility* of TSP as a fourtuple:

$$F =< \mathcal{C}^T, f_C; \mathcal{C}^E, f_E > , \tag{2.25}$$

where

$\mathcal{C}^T$ is the *deadline* and *release time constraint*,
f_C is the target completion time function,
$\mathcal{C}^E$ is the *resource* (energy) *constraint*, and
f_E is the energy consumption function.

2.2.1.3.1. Time Constraints $(\mathcal{C}^T, f_C)$. Two time constraints define a window of opportunity during which a target must be addressed. First, targets may not be processed before they are "available". A target may be unavailable, for instance, before breaking through a cloud cover. The battle manager may have several additional reasons why a given target should not be addressed by a platform prior to some time. Second, targets must be processed by a given time called the *deadline*.

Mathematically, these time constraints can be simply stated as a

$$\textit{completion constraint}: \quad t_{c\pi(i)} \leq d_{\pi(i)}, \quad i = 1, \ldots, |I|\,, \text{ and a} \tag{2.26}$$

$$\textit{release constraint}: \quad t_{c\pi(i)} - t^D_{\pi(i)} \geq r_{\pi(i)}\,, \tag{2.27}$$

where $d_{\pi(i)}$ is the deadline for target $T_{\pi(i)}$, and $r_{\pi(i)}$ is its *release time*. Observe that the release constraint is implicitly included in the expression for completion time (see Eq. (2.30)).

Given the definition of sequences and permutations presented earlier, the target completion time function is a mapping

$$f_C : \mathcal{T} \times \Pi \times R^+ \times R^+ \rightarrow R^+ \tag{2.28}$$

that assigns to each target $T_i \in \mathcal{T}$, permutation $\pi \in \Pi$, dwell time $t^D \in R^+$, and release time $r \in R^+$ a *completion time* $f_C(T_i, \pi, t^D, r)$.

Note that completion times also implicitly depend upon the platform $\mathcal{D}$, the environment $\mathcal{E}$, and the sensor set $\mathcal{S}$. We defer to later chapters the explicit derivation of this dependence; in this chapter we limit our discussion to the situation where $\mathcal{D}$, $\mathcal{E}$, and $\mathcal{S}$ are assumed fixed.

Consider now the ordered set of completion times $CT_\pi =< t_{c\pi(1)}, \ldots, t_{c\pi(i)}, \cdots, t_{c\pi(|I|)} >$, where $t_{c\pi(i)}$ is the completion time of target $T_{\pi(i)}$ under permutation π. Then f_C is a recursive function defined as follows for all $\pi \in \Pi$:

$$t_{c\pi(1)} = f_C(t_{c\pi(1)}, \pi) = r_{\pi(1)} + t^D_{\pi(1)}\,, \tag{2.29}$$

where

$r_{\pi(1)}$ is the release (availability) time of the first target $T_{\pi(1)}$, and

$t^D_{\pi(1)}$ is the dwell time of the first target $T_{\pi(1)}$.

Inductively: for $i = 1, 2, \ldots, |I|$,

$$t_{c\pi(i)} = f_C(t_{\pi(i)}, \pi)$$

$$= \max\left\{r_{\pi(i)}, t_{c\pi(i-1)} + t^R_{\pi(i)}\right\} + t^D_{\pi(i)} \ , \tag{2.30}$$

where

$r_{\pi(i)}$ is the release time of target $T_{\pi(i)}$.

$t^R_{\pi(i)}$ is the retarget time from $T_{\pi(i-1)}$ to $T_{\pi(i)}$

$t_{c\pi(i-1)}$ is the completion time of $T_{\pi(i-1)}$.

$t^D_{\pi(i)}$ is the dwell time of $T_{\pi(i)}$.

Note how release times can determine completion times: regardless of the completion time of the previous target and the retarget time to the current target, processing of the current target may not start until it is ready or *available.*

The calculation of dwell times $t^D_{\pi(i)}$ depends upon physical parameters and variables that are discussed in a later section; retarget time computation is discussed next.

2.2.1.3.1.1. Retarget Times. The retarget time $t^R_{\pi(i)}$ from Target $T_{\pi(i-1)}$ to Target $T_{\pi(i)}$ is determined by the state of the two targets at the completion time $t_{c\pi(i-1)}$ of $T_{\pi(i-1)}$ and by the dynamical properties of the DEW retargeting hardware. A typical retarget timeline is shown in Fig. 2.3 where a familiar second-order behavior of the platform hardware is portrayed.

In response to an angular retarget step input $u(t)$, the time required to settle to within ϵ of the location of $T_{\pi(i)}$ is

$$t^R_{\pi(i)} = t_{sat} + t_{react} + t_{rise} + \tau_{set} \ , \tag{2.31}$$

where the meaning of the variables is obvious (Kuo 1975). More conventionally,

$$t^R_{\pi(i)} = t_{sat} + t_{set} \ , \tag{2.32}$$

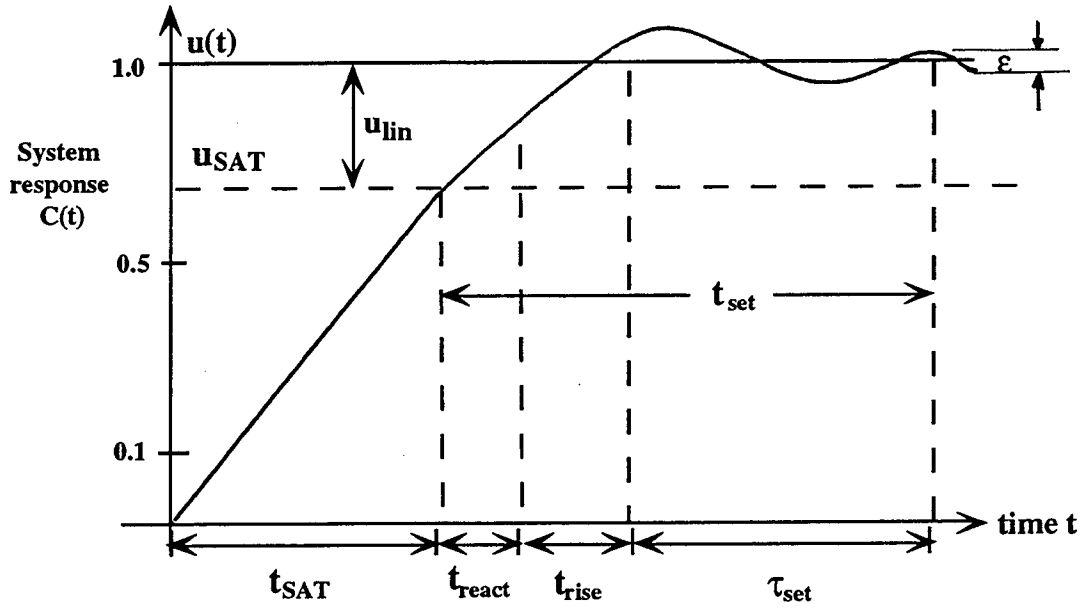

Figure 2.3. Second-order non-linear approach to estimating DEW retarget time showing unit STEP response time.

where

t_{sat} is the *saturation time*, and $t_{sat} = \dfrac{u(t) - u_{lin}}{\dot{c}_{max}}$,

$\dot{c}_{max}$ is the maximum slew rate,

u_{lin} is the maximum input before the onset of saturation, i.e., the *maximum linear response input*,

t_{set} is the *settling time* and $t_{set} = \dfrac{|ln(\epsilon)|}{\sigma\omega_n}$,

δ is the *damping constant*, and

ω_n is the *critical frequency*.

More compactly,

$$t^R_{\pi(i)} = \frac{max\{u - u_{lin}, 0\}}{\dot{c}_{max}} + \frac{|ln(\epsilon)|}{\sigma\omega_n} \,. \tag{2.33}$$

A slight complication arises due to the fact that a DEW is a composite structure consisting of *three* major mechanical components: the main body (MB), the forebody (FB), and the fast steering (FS) subsystem. When angular displacement commands are sufficiently large, all three subsystems may have to be slewed, and the retarget time

is then also a composite function

$$t^R_{\pi(i)} = max\left\{t^R_{FS\pi(i)},\ t^R_{FB\pi(i)}U\left(\theta_i - \frac{\theta^{FS}_{max}}{2}\right),\quad t^R_{MB\pi(i)}U\left(\theta_i - \frac{\theta^{FB}_{max}}{2}\right)\right\}, \tag{2.34}$$

where

$t^R_{FS\pi(i)},\ t^R_{FB\pi(i)},\ t^R_{MB\pi(i)}$ are the fast steering, forebody, and main body retarget times derived as in Fig. 2.3,

$$U(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases},$$

θ_i is the angular distance from $T_{\pi(i-1)}$ to $T_{\pi(i)}$, and

$\theta^{FS}_{max},\ \theta^{FB}_{max}$ are the maximum displacements of the fast steering and forebody subsystems, respectively.

The step function $U(x)$ is important here since no forebody or mainbody slew is required unless the input command θ_i exceeds the fast steering (respectively the forebody) motion limits.

Considering that all the targets are moving and that a scheduling algorithm must estimate the state of targets some time far into the future, the computation of the angular separation θ_i deserves further comment. Referring to Fig. 2.4 where a DEW platform D is located at position $D_E = (D_{E,x},\ D_{E,y},\ D_{E,z})$ in an earth-centered coordinate system (x_E, y_E, z_E) as shown, the angular separation between T_{i-1} and T_i in platform coordinates $(x_P,\ y_P,\ z_P)$ is

$$\theta_i = \cos^{-1}\left(\frac{X_{i-1,P}\cdot X_{i,P}}{\parallel X_{i-1,P} \parallel \parallel X_{i,P} \parallel}\right) = \cos^{-1}\left(e_{i-1,P}\cdot e_{i,P}\right), \tag{2.35}$$

where $e_{i-1,P}$ and $e_{i,P}$ are the corresponding unit vectors. Relating this to the earth-centered coordinate system,

$$X_{i-1,p} = R(X_{i,E} - D_E), \tag{2.36}$$

where

R is the rotation matrix from earth to platform coordinates,
$X_{i,E}$ is the position of T_i in earth coordinates, and
D_E is the position of D in earth coordinates,

and similarly for T_{i-1}.

In practice, we use the distance $\epsilon^P_i = e^P_{i-1} - e^P_i$ as a guide to estimate θ_i, and we define

$$\theta_i = \begin{cases} \parallel \epsilon^P_i \parallel rad & ,\ \parallel \epsilon^P_i \parallel \leq 0.1745 \\ \cos^{-1}(e^P_{i-1}\cdot e^P_i)\ rad & ,\ \parallel \epsilon^P_i \parallel > 0.1745 \end{cases}. \tag{2.37}$$

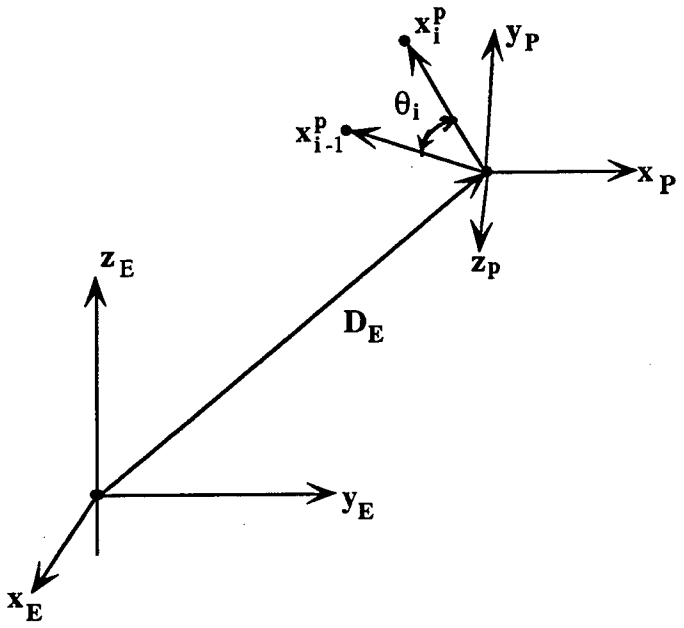

Figure 2.4. Computing the angular separation θ_i between targets T_{i-1} and T_i in earth coordinates.

To predict the position of a target Δt time units into the future from some reference time t_0 (initiation or handover time), we extrapolate initial positions (x_0, y_0, z_0) and initial velocities ($\dot{x}_0$, $\dot{y}_0$, $\dot{z}_0$) along a straight line trajectory or along a constant-radius circle centered at the earth, as discussed in Section 2.2.1.1.

2.2.1.3.1.2. *Dwell Times t^D.* The target interaction physics from which dwell times are derived are very phase-dependent since the interaction mechanisms involved in the boost phase differ significantly from those in the midcourse phase. Generalization is thus not very useful, and we defer to Chapters 3 and 5 any detailed derivations of dwell times for the boost phase and the midcourse discrimination phase, respectively. An important fact used in the remainder of this chapter, however, is that there is one-to-one relationship between dwell times and miss probabilities (nondiscrimination probabilities in the midcourse phase), so that, once miss probabilities (p_M) are specified, dwell times are uniquely determined.

2.2.1.3.2. Resource Constraints ($\mathcal{C}^E$, f_E). Various limited resources are consumed by a platform during an engagement, and a careful resource management strategy must be included in the target scheduling algorithm to avoid reaching resource limits at mission-critical times. To accomplish this, we use a local *resource consumption function*

$$E^\circ : \mathcal{T} \times \Pi \times R^+ \rightarrow R^+ , \tag{2.38}$$

which assigns to each target $T_i \in \mathcal{T}$, permutation $\pi \in \Pi$, and dwell time $t_i^D \in R^+$, the quantity of energy $E^\circ(T_i,\ \pi, t_i^D)$ consumed during the retargeting from $T_{\pi(i-1)}$ to $T_{\pi(i)}$ and during the dwell time t_i^D on target $T_{\pi(i)}$.

The total consumption of resources is recursively described by the *cumulative resource function*

$$f_{E_i} = \sum_{j \le i} E^\circ(T_j,\ \pi, t_j^D)\ , \tag{2.39}$$

recursively defined by

$$\begin{aligned} f_{E_i} &= f_{E_{i-1}} + E^\circ(T_j,\ \pi,\ t_j^D) \\ &= f_{E_{i-1}} + E^R(t^R_{\pi(i)}) + E^D(t^D_{\pi(i)})\ , \end{aligned} \tag{2.40}$$

where

$E^R(t^R_{\pi(i)})$ is the quantity of resources consumed during retargeting time $t^R_{\pi(i)}$.

$E^D(t^D_{\pi(i)})$ is the quantity of resources consumed during dwell time $t^D_{\pi(i)}$.

The energy constraint is now simply:

$$\text{For all } i = 1, \ldots,\ |I|,\ \ f_{E_i} \le E_{max}\ , \tag{2.41}$$

where E_{max} is the total amount of resources available.

2.2.1.4. The Optimization Criterion H. "Expected risk" has been found in practice to best reflect the degree and uncertainty of loss in complex large-scale decision problems not dominated by rare events (Ferguson 1967, Berger 1980). As we mentioned in Section 2.2.1, risk may be defined using an *optimization criterion*

$$H =< L, \mathcal{P}_L, \Sigma\ E >\ , \tag{2.42}$$

where

L is the *loss function.*
$\mathcal{P}_L$ is the *leakage probability* function.
$\Sigma\ E$ is the *summation-expection composition operator on* $\mathcal{L}$, producing for each loss sequence $L(w, a) \in \mathcal{L}$, the sum of all the expected rejection and miss losses associated with the terms of the sequence with respect to the probability measure $\mathcal{P}_L$.

The loss function (L) was defined earlier in Section 2.2.1.2.2. The probability that a target leaks through ($\mathcal{P}_L$) is essentially determined by the lethality of the DEW, the hardness of the target, the environment and time of the engagement, the probability

that the target is correctly identified, the dwell time allocated to the target, and deadline constraints. As before, we assume that the DEW (D), the environment (E), the sensors (S), and the constraints ($\mathcal{C}$) are fixed during a single target scheduling cycle, and all probability assertions and calculations are thus conditional to D, E, S, and $\mathcal{C}$.

We assume that a target leaks through if, and only if, it is rejected (ignored) by the target scheduler, or it is missed by the DEW, or it is misidentified. The physical conditions that give rise to these three forms of leakage are represented in the same space Ω (states of nature) discussed in Sections 2.1 and 2.2.1.1. The probability of leakage p_L depends upon these three leakage modes as follows.

Let $\overline{ID}$, R, and M stand for target misclassification, rejection, and miss, respectively. Then

$$\begin{aligned} p_L &= \text{prob}\,(\overline{ID} \vee R \vee M) \\ &= 1 - \text{prob}\,(ID \wedge \overline{R} \wedge \overline{M}) \\ &= 1 - \text{prob}\,(\overline{M}\,|_{ID \wedge \overline{R}})\text{prob}\,(ID)\text{prob}(\overline{R}) \\ &= 1 - (p_{\overline{M}})\,|_{ID \wedge \overline{R}})p_{ID}p_{\overline{R}} \end{aligned} \tag{2.43}$$

where $p_{\overline{M}} = \text{prob}\,(\overline{M})$, $p_{ID} = \text{prob}\,(ID)$, and $p_{\overline{R}} = \text{prob}(\overline{R})$.

To separate the contributions to p_L made by the miss probability p_M and the rejection probability p_R, Eq. (2.43) may be rewritten as

$$p_L = p_R + \left[1 - (p_{\overline{M}}\,|_{ID})p_{ID}\right]p_{\overline{R}}\,. \tag{2.44}$$

Equation (2.43) shows that a target leaks through if, and only if, it is rejected, misidentified, or missed when it is correctly identified and not rejected.

The target ID probability p_{ID} is derived from a Bayesian classification process that will be discussed in a future report (Corynen 1993). Because target rejection is a complex heuristic optimization process, no analytical expression for the rejection probability p_R can be derived.

The conditional miss probability ($p_M\,|_{ID \wedge \overline{R}}$) is obtained from a statistical comparison between weapon lethality and target hardness, as follows:

$$p_M(T_{\pi(i)}) = \text{prob}\left(H_{\pi(i)}(Q^T_{\pi(i)},\, t_{c\pi(i)},\, \omega^T) > L(Q^D, t_{c\pi(i)},\, \omega^D,\, t^D_{\pi(i)})\right)\,, \tag{2.45}$$

where

$H_{\pi(i)}$ is the *hardness* of target $T_{\pi(i)}$ whose state is $Q^T_{\pi(i)}$ at completion time $t_{c\pi(i)}$, when the random state of nature (environment) of the target is ω^T.

L is the *lethality* of the weapons platform D whose state is Q^D at time $t_{c\pi(i)}$, when the dwell time is $t^D_{\pi(i)}$ and the platform state of nature is ω^D.

Using Eq. (2.44), we may summarize all this in terms of a general objective function, as follows:

The total leakage risk is

$$R_L = \sum_{i=1}^{|I|} V_i p_{L_i} = \sum_{i=1}^{|I|} V_i p_{R_i} + \sum_{i=1}^{|I|} V_i \Big[1 - (p_{\overline{M}} \mid_{ID_i}) p_{ID_i}\Big] p_{\overline{R}_i}$$

$$= \mathcal{R}_r + \mathcal{R}_m \tag{2.46}$$

where

$\mathcal{R}_r$ is the *rejection risk*, and $\mathcal{R}_r = \sum_{i=1}^{|I|} V_i p_{R_i}$,

$\mathcal{R}_m$ is the *miss risk*, and $\mathcal{R}_m = \sum_{i=1}^{|I|} V_i [1 - (p_{\overline{M}} \mid_{ID_i}) p_{ID_i}] p_{\overline{R}_i}$,

L_i is the loss incurred if Target i leaks through, and
p_{L_i} is the probability that Target i leaks through.

The expected scheduling leakage risk can now be expressed more explicitly as the sum of a *rejection risk* $\mathcal{R}_r$, *an identification risk* $\mathcal{R}_{ID}$, and a *miss risk* $\mathcal{R}_m$, as follows. Let R_i be the event "Target T_i is rejected," $\overline{R}_i$ its complement, and let $p(L_i \mid_E)$ be the probability that Target T_i leaks through condititional to the occurrence of event E. Then the expected loss for a single target T_i is

$$L_i p(L_i) = L_i p(L_i \mid_{R_i}) p(R_i) + L_i p(L_i \mid_{\overline{R}_i}) p(\overline{R}_i)$$

$$= L_i p(R_i) + L_i p(Mi) p(\overline{R}_i) \,, \tag{2.47}$$

where $p(M_i) = p(L_i \mid_{\overline{R}_i})$, the probability that T_i is missed if its interception is attempted. This probability will also be called the *miss probability.*

Successful target interception strongly depends upon the target identification power of the sensors observing an engagement. While target rejection is assumed unaffected by the ID process, we assume that incorrectly identified targets leak through. If ID_i is the event "Target T_i is correctly identified", and $\overline{ID}_i$ is its complement,

$$L_i p(L_i) = L_i p(R_i) + L_i p(M_i \mid_{\overline{ID}_i}) p(\overline{ID}_i) p(\overline{R}_i)$$

$$+ L_i p(Mi \mid_{ID_i}) p(ID_i) p(\overline{R}_i) \,.$$

Since $p(Mi\ |_{\overline{ID}_i}) = 1$,

$$L_i p(L_i) = L_i p(R_i) + L_i p(\overline{ID}_i) p(\overline{R}_i) + L_i p(Mi\ |_{ID_i}) p(ID_i) p(\overline{R}_i)\ , \tag{2.48}$$

and the total leakage risk for a sequence π is simply the summation of Eq. (2.46) over i (slightly rewritten):

$$\begin{aligned}\mathcal{R}_L(d) &= \sum_{i=1}^{|I|} L_i p_r(T_{\pi(i)}) + \sum_{i=1}^{|I|} L_i p_{\overline{ID}}(T_{\pi(i)})(1 - p_r(T_{\pi(i)})) \\ &\quad + \sum_{i=1}^{|I|} L_i p_m\Big(T_{\pi(i)}\ |_{ID}\ \Big) p_{ID}(T_{\pi(i)})(1 - p_r(T_{\pi(i)})) \\ &= \mathcal{R}_r(d) + \mathcal{R}_{ID}(d) + \mathcal{R}_m(d)\end{aligned} \tag{2.49}$$

where $d = (d_D,\ d_\pi,\ d_r)$, and the meaning of the symbols is obvious from earlier definitions.

2.2.2. The Optimization Problem

In this section, we present a general discussion of the schedule optimization problem. First, we introduce the problem in general terms. Then we outline the major steps of our approach.

2.2.2.1. Introduction. The Target Scheduling Problem (TSP) consists of processing a collection of targets in accordance with a schedule that minimizes the target leakage risk, subject to deadline, release time, and energy constraints. A target leaks through either because it is *rejected* by the platform scheduler, or it is missed by the DEW during processing. Only when a deadline cannot be met must one or more targets be rejected. From an expected risk perspective, rejection losses are worse than miss losses because they occur with certainty when deadlines cannot be met, whereas miss losses are weighted by their probability of occurrence. Consequently, obeying deadlines is the major objective of our scheduling method, and the resulting algorithm was called the Deadline Driven Target Scheduling Algorithm (DDTS).

Observe that the absence of deadlines does not eliminate leakage risk altogether, even though the schedules may allocate arbitrarily large dwell times to the targets. An approach dedicated to avoiding tardiness at all cost may not do well in situations where deadlines are loose or soft (low value of deadline hardness factor α). For some practical scenarios, in fact, waiting for a more propitious geometry may considerably reduce the risk, and any method designed to process all targets as quickly as possible will be suboptimal for such cases.

Recall the mathematical setting developed in earlier sections. Given a set $\mathcal{T} = \{T_i : i = 1,\ \ldots,\ |\ I\ |\}$ of targets whose values are $V_i, i = 1, \ldots, |\ I\ |$, a target observation

process $\mathcal{X} = (X_1(\omega), \ldots, X_n(\omega))$ on the states of nature $\omega \in \Omega$, with values $x^n \in \mathcal{R}^n$, a set $\mathcal{A}$ of admissible actions, and a decision function $d : \mathcal{R}^n \rightarrow \mathcal{A}$. For every observation or scenario $x^n(\omega) \in \mathcal{R}^n$, d produces a decision or action $d(x^n(\omega)) = (t^D,\ \pi,\ \mathcal{T}_r) = a \in \mathcal{A}$. Each decision a incurs a loss $L(\omega, d(x^n(\omega))) = < V_{\pi(i)} \chi^L_{\pi(i)}) : i = 1,\ \ldots,\ | I | >$ when the true state of nature is ω. The TSP is to find a decision function d^* that minimizes the total expected leakage risk specified by Eq. (2.49), and subject to the constraints C^T and C^E introduced earlier.

The apparent simplicity of the risk function $\mathcal{R}_L$ is quite deceptive because the TSP is considerably more complicated than familiar traveling salesman or job shop scheduling problems. First, due to platform and target motions and the associated time-variance of parameters like DEW lethality and target vulnerability, the TSP is a *dynamic* problem. Potentially every permutation $\pi \in \Pi$ may have to be tested. Second, deadlines and release times may require that carefully selected targets be allowed to leak through since time window constraints may not be met otherwise. The optimization procedure may therefore be required to test every subset $\mathcal{T}_r \subset \mathcal{T}$ of targets to find those targets which should be ignored. Finally, since dwell times strongly affect the target completion times and probabilities of leakage, their selection is an important part of the optimization procedure. Because each target typically receives a different dwell time, every dwell time vector $t^D \in \prod_{i \in I} R_i$ may have to be tried. Observe that the sensor architecture and the torque shaping commands are not currently subject to optimization in our framework.

Relating this to the canonical optimization problem of Section 2.1, the action space $\mathcal{A}$ thus consists of an uncountably large collection of triples $(t^D,\ \pi,\ \mathcal{T}_r)$, each of which may have to be tested to find the minimum leakage risk. Since, for any choice $(t^D,\ \pi,\ \mathcal{T}_r)$ and real number R^o, the validity of $\mathcal{R}(t^D,\ \pi,\ \mathcal{T}_r) \leq R^o$ can be "recognized" in polynomial time, TSP is clearly in the class of Nondeterministic Polynomial (NP) Problems.* But since the standard traveling salesman problem polynomially transforms* to TSP, all problems in NP transform polynomially to TSP, and the Target Scheduling Problem is also NP-complete. This is not surprising, since Savelsberg (1984) showed that even finding a *feasible* solution to the non-stochastic and static traveling salesman problem with time windows is NP-hard. The $O(n^2)$ spanning tree problems also become NP-hard when time windows are introduced (Solomon 1986).

Conventionally, complex problems like the TSP in this report are addressed with Monte Carlo simulation, queueing networks, or Petri Nets. Such methods are excluded in most real-time situations, however, and only heuristic optimization approaches will succeed in estimating the minimum leakage risk within acceptable bounds of accuracy and algorithmic complexity. The completion time function and the energy consumption function are of little assistance since they are both recursive. Because the objective

* Papadimitriou and Steiglitz (1982).

function $\mathcal{R}$ is therefore recursive as well, its internal structure cannot easily be exploited for computational purposes.

2.2.2.2. Optimization Approach. There are no ready answers to the TSP, not only because the problem is NP-hard, but also because it involves real-time computational challenges peculiar to target scheduling. Many useful results in scheduling theory and discrete optimization are available to solve related problems or special cases of TSP, and every reasonable effort was made in this report to exploit past work by assuming that limiting cases of TSP converged to problems for which analytic solutions do exist. This minimized the risks associated with any inaccuracies or unsuspected computational traps.

Our real-time approach minimizes the *Conditional Risk* $\mathcal{R}_L\,(d_D, d_\pi, d_\tau\,|_{x^n})$ given the observations x^n, the standard extensive form of Bayesian analysis where posterior loss is minimized conditional to the data.* This is equivalent to minimizing the conditional risk for every $x^n \in R^n$ and, since a sufficient condition for minimizing an integral is to minimize its integrand over its entire range, it avoids the integration required to calculate the overall *unconditional risk*,

$$\mathcal{R}_L(d_D,\ d_\pi,\ d_\tau) = \int_{R_n} \mathcal{R}_L((d_D,\ d_\pi,\ d_\tau)\,|_{x^n})\ f(x^n)\ dx^n. \tag{2.50}$$

In this report, we shall thus consider only the *conditional risk*, expressed in terms of the actions resulting from the decision triple d and the observation x^n:

$$\begin{aligned}\mathcal{R}_L\Big((d_D,\ d_\pi,\ d_\tau)\,|_{x_n}\Big) &= \mathcal{R}_L\Big(d_D(x^n),\ d_\pi(x^n),\ d_\tau(x^n)\Big) \\ &= \mathcal{R}_L\Big(a_D,\ a_\pi,\ a_\tau\Big)\ .\end{aligned} \tag{2.51}$$

Observe that x^n does not provide measurements on the entire sample space Ω, and some probability computations are still required. The space Ω^T underlying the target parameter vector π^T (Eq. (2.7)), for instance, cannot be sampled during algorithm execution since parameters like Target Hardness are unobservable in real time. This also applies to the platform sample space Ω^D.

The first step in minimizing multi-dimensional functions is to look for mathematical structure such as convexity, monotonicity, and separability. Considering separability first, no separation of the risk function $\mathcal{R}_L$ is possible because of the strong interdependence of the decision variables. Considering a_τ next, we reject targets on the basis of Marginal Risk Reduction whenever deadlines cannot be met. Therefore $\mathcal{R}_L(a_D,\ a_\pi,\ a_\tau)$ is monotonic only in the total value of the set $\mathcal{T}_\tau$ of rejected targets and not in a_τ itself since the a_τ's are *sets* with no strict ordering.

* Ferguson (1967); Berger (1980); Duda and Hart (1973); Jain (1989).

Next, consider dwell time allocations a_D. As dwell times are increased, the probability of successful target processing increases, and conversely. But each target could conceivably be allocated a different dwell time or leakage probability p_L, hence t^D or a_D is actually an n-dimensional vector whose optimization would be impractical in real time. To explore the possibility of allocating just a single common dwell time to all targets, we note that encouraging tests have indicated that leakage probabilities tend to bunch around a common value during such as optimization. One explanation for this bunching effect is that our rejection schedule (see Eq. [2.52]) severely penalizes targets with a high t^D value, but the expected risk criterion also strongly discourages retaining targets with a low p_K value.

While the computational benefits of assigning a common p_K to all targets are enormous (the dimension of the optimization problem is reduced by a factor of $|I|$), forcing an increase in p_K for all targets may significantly increase their dwell times and may cause additional target rejections, thereby obtaining only a suboptimal solution. Consider for instance a high-valued target that is close to the DEW, requiring considerably less dwell time than the other targets to achieve a given p_K. From a marginal benefit perspective it would be desirable to allow increasing the p_K of the valuable target, while retaining—or even decreasing—the p_K of the other targets, something that the DDTS algorithm does not allow. Note however that the entire scheduling process is recursive: targets rejected at one time will reappear again as unprocessed targets at another time and will be allocated an appropriate amount of dwell time as soon as high-priority targets have been processed. Therefore, in spite of the remote possibility of producing a suboptimal schedule, the dwell decision function d_D assigns to each target an equal p_L. As a final comment about the structure of d_D, we show below that, except for minor perturbations, $\mathcal{R}_L(a_D,\ a_\pi,\ a_\tau)$ is also convex in a_D.

No natural ordering of the actions $\alpha_\pi \in \Pi$ exists for TSP Thus $\mathcal{R}_L$ cannot be monotonic or convex in α_π, and a combinatorial heuristic will be required to find the optimal sequence π^*.

A simplified flowchart of the DDTS algorithm is shown in Fig. 2.5. Measurement inputs are obtained from on-board or from external sensors. Other inputs include external handovers, constraints, and objectives. The measurements are samples $x^n \in R^n$ of the measurement process $\mathcal{X}$ discussed in detail in Section 2.2.1.1. Constraints are specified by the *feasibility* $F =< \mathcal{C}^T,\ f_C;\ \mathcal{C}^E, f_E >$ defined in Section 2.2.1.3, and objectives are specified via the optimization criterion $H =< L, \mathcal{P}, \Sigma E >$ discussed in Section 2.2.1.4. One input not discussed thus far is the *scheduling horizon* t_s^{max}, the maximum time into the future for which targets must be scheduled. This input parameter is needed for two major reasons. First, the complexity (i.e., running time) of DDTS is a polynomial in the quantity $|I|$ of targets, increasing significantly as $|I|$

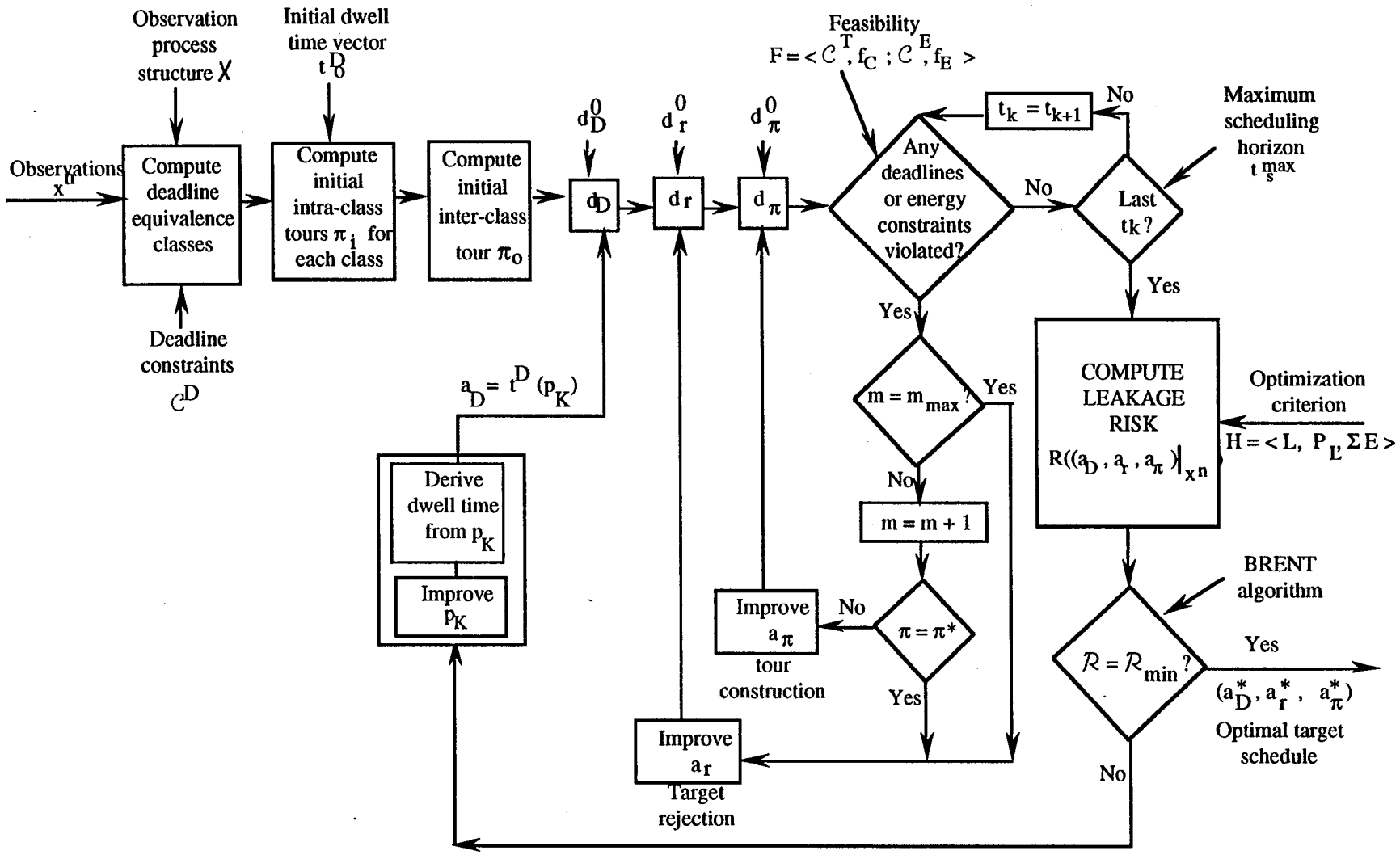

Figure 2.5. Simplified flowchart of the Deadline-Driven Target Scheduling (DDTS) algorithm.

is increased. Second, the reliability of predictions or estimates decreases considerably as the horizon is increased. Although additional experiments may be needed to determine a best value for t_s^{max}, currently a value of 100 seconds is used.

In our decision-making setting, DDTS is viewed as a decision maker that takes observations $x^n(\omega)$ into actions $a = (a_D,\ a_\pi,\ a_r) \in \mathcal{A}$, thereby incurring a loss $L(\omega, a)$ (see Section 2.2.1.2.). Only actions a_D and a_π produce an output—dwell time t^D and target sequence π, respectively. When platform jerking is a serious problem, specific retargeting states are produced to minimize jerk, but this is not considered in the first version of the algorithm. (See Section 3.4 for a derivation of smooth command schedules.)

Decisions are made to minimize the overall objective function $\mathcal{R}_L(d_D,\ d_\pi,\ d_r)$ of Eq. (2.48), subject to the feasibility constraints F. The complexity of the optimization problems rules out an analytical solution, and our heuristic approach consists of three nested loops, as shown in Fig. 2.5. Before executing these loops, the algorithm is initialized by calculating the deadline equivalence classes and by constructing an initial tour through these classes. Two targets belong to the same class if they have the same deadline and release time. Whenever two targets have the same deadline but different release times, the one with the earliest release time is ranked first.

The initial tour π_0 through all the targets is found in two steps. First, a shortest tour *within* each of the equivalence classes is found, one for each class. These are the *internal tours.* Then a *global tour* is constructed where classes are considered as single points and class elements are included in accordance with their internal tours. Tour construction is done using the Farthest Insertion Algorithm*, and, in global tours, targets in an equivalence class precede those in another class if, and only if, the deadlines of the former precede those of the latter. Observe, however, that this initial ordering of targets may later be disturbed when tour corrections are needed to meet deadlines.

The execution of the three loops may be viewed as occurring sequentially, with sequence selection first, then target rejection, then dwell time optimization. Starting with an initial dwell time t_0^D and tour π_0, the first operation is to test completion times t_c against deadlines. If no deadlines are missed up to t_s^{max} (horizon limit), no improvements to π_0 are needed and no targets are rejected, and all that remains is to optimize dwell time t^D. If one or more deadlines are missed at some time t_k, tour improvement methods such as 2-opt* are brought into action until no more deadlines are missed, until a maximum number n_π of tour improvement attempts have been made, or until an optimum tour π^* has been found. During tour improvement, the rejection risk $\mathcal{R}_r$ is minimized by sufficiently reducing the completion time of important targets by varying the permutation $\pi \in \Pi$. Observe that not *all* completion times need to be minimized, only those that would lead to important targets missing their deadlines. In all but the tightest scheduling scenarios, only a few corrections need to be made since

* Lawler et al. (1985); Norback and Love (1977).

few targets would miss their deadlines. Note also that the rejection risk will often be "flat" in the π variable because many sequences will yield the minimum risk $\mathcal{R}_L^*$.

If some deadlines are still missed, some targets must be rejected. This is accomplished by rejecting targets whose removal produces the largest marginal risk reduction, using the *rejection criterion*

$$r = \frac{\alpha \cdot p_{ID}^* \cdot L^*}{t^{D^*} + \Delta t^R}, \tag{2.51}$$

where

α is the *deadline hardness* ($\alpha \in [0,1]$).
p_{ID}^* is the probability of correctly identifying the target type, conditional to deciding its type is θ^*.
L^* is the expected target loss conditional to θ^*.
t^{D^*} is the target dwell time conditional to θ^*.
Δt^R is the retarget time earned by skipping the target.

While the obvious strategy is thus to skip a target whose ratio is *least*, note the interesting circumstance where such a least target precedes an "early" target, one where waiting occurs. Recall that the processing of a target cannot be started before its release time. Hence there is no point in skipping a target that precedes an early target since that will simply increase the waiting time at the early target, and this consideration is included in the DDTS algorithm. While there is no analytical proof that this rejection scheme is optimal, two important limiting cases have actually been shown to be optimal. With the *quantity* of late targets as a risk measure, Kise, Ibaraki, and Mine (1978) have shown for the static case that the numerator of Eq. (2.51) is the best rule. In the standard "shortest-job-first" approach, Smith (1956) and others* have verified that, when all tasks (targets) have the same value, then r of Eq. 2.50 is the also the best rule. Additional work by Lawler (1971) and others contains significant special cases where the ratio of value to time is an optimal ranking criterion.

When the optimal tour and rejection have been found, the outer loop is re-entered, and a new vector of dwell times is attempted, until the overall leakage risk is minimized. Recall that dwell time vectors t^D are obtained from common values of p_L for all targets using a relationship whose boost phase version is derived in Section 3.3.1.4. This optimization process can be explained by using just two targets whose values are V_1 and V_2 and whose deadlines are $d_1 < d_2$, respectively, as shown in Fig. 2.6, where $p_K = 1 - p_L$.

As p_K is increased from 0, the total risk decreases as $(1 - p_K)\ (V_1 + V_2)$ until the first deadline d_1 is exceeded, and the risk jumps to $V_1 + (1 - p_K)V_2$. Additional increases in p_K further decrease that risk until the second deadline is violated, and both targets leak

* Abdul-Razaq et al. (1990); French (1982); Baker (1974); Vickson (1980a,b).

through. When the first deadline is violated, the risk therefore jumps by $V_1 p_K$ before dropping to its minimum value $V_1 + (1 - p_K)V_2$ associated with the maximum dwell time for both targets—i.e., when $t_{c2} = d_2$.

For the general multi-target case, the risk function therefore may have several local minima, technically as many as there are targets, and this could severely complicate the optimization process. But this is a potential problem only for sparse threats, since for large threats the jumps are relatively very small and can be ignored with most optimization algorithms by setting the tolerance or error coefficient sufficiently high, as we have done using the Brent Algorithm (Press et al. 1988; Brent 1973). When threats are in fact sparse, deadlines are also less stringent, and there is thus no significant approximation problem in either a target-rich or a target-poor environment.

To conclude our discussion on optimization, recall an earlier discussion of scenarios or threats for which deadlines were very loose or very soft, and where an early processing of the targets incurred a large risk due to an inconvenient threat geometry in which threats are far away from the DEW platform. In cases such as this, delaying any action can often improve the overall leakage risk, and such delays can be introduced by simply increasing the common p_K, because that also increases the dwell times and the resulting completion times. This may occassionally lead to minor violations of the energy constraints; however, in such a case, a waiting time should be introduced, but this is not done in the current version of DDTS.

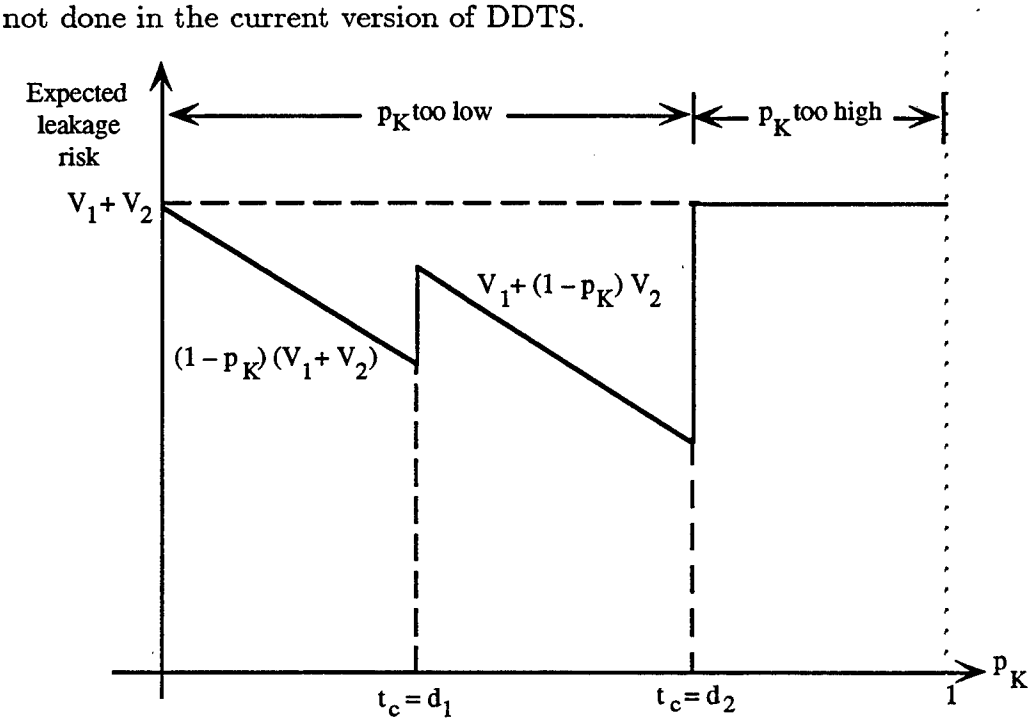

Figure 2.6. Risk decreases with increasing p_K until the dwell time causes target tardiness.

3. Scheduling Theory for the Boost Phase

In this chapter, we address the Target Scheduling Problem in greater detail by focusing on the Boost Phase. In Chapter 5, the boost phase solution will be extended to the more complex midcourse discrimination problem.

When using the objective function of Eqs. (2.49) and (2.50), and as illustrated in Fig. 2.5, the problem suggests a decomposition into three subproblems:

1. Finding an optimal sequence $\pi^* \in \Pi$.
2. Rejecting targets.
3. Computing an optimal dwell time vector $t^D \in \prod_{j-1}^{n} R_j$.

While it is clear that these subproblems are generally not independent, such a decomposition has important benefits because significant special cases of the TSP often reduce to just one of these subproblems. The strong resemblance to standard problems in combinatorial optimization was an additional motivation to develop connections to the theoretical optimization business in order to benefit from the considerable work of others. We therefore relate each subproblem to the combinatorial optimization literature to show how others have addressed similar problems. Then we show how the solution to each subproblem is combined into a solution to the entire target scheduling problem.

Considering the important trade-off between speed and accuracy, perhaps a more important reason for this three-part decomposition is the admission that we have been unable to derive tight upper and lower bounds for the accuracy of the DDTS algorithm. Given that bounds *are* available for certain versions of its subproblems, the algorithm was designed to agree with those bounds on these subproblems, and we are currently pursuing an indirect approach where global bounds are derived from these "subbounds" and from structural information relating the subproblems.

To provide a slightly different perspective for discussion purposes, the DDTS algorithm may be viewed as consisting of two major nested optimization loops, a time optimization loop and a leakage probability minimization loop, as shown in Fig. 3.1. This figure is a further simplification of Fig. 2.5 and shows the inner loop as a local time optimization loop where both the target *sequence* and the target *rejection* set are optimized. In the outer loop, the leakage probabilities $p_L = 1 - p_K$ are minimized via the allocated target dwell times $t^D(p_K)$.

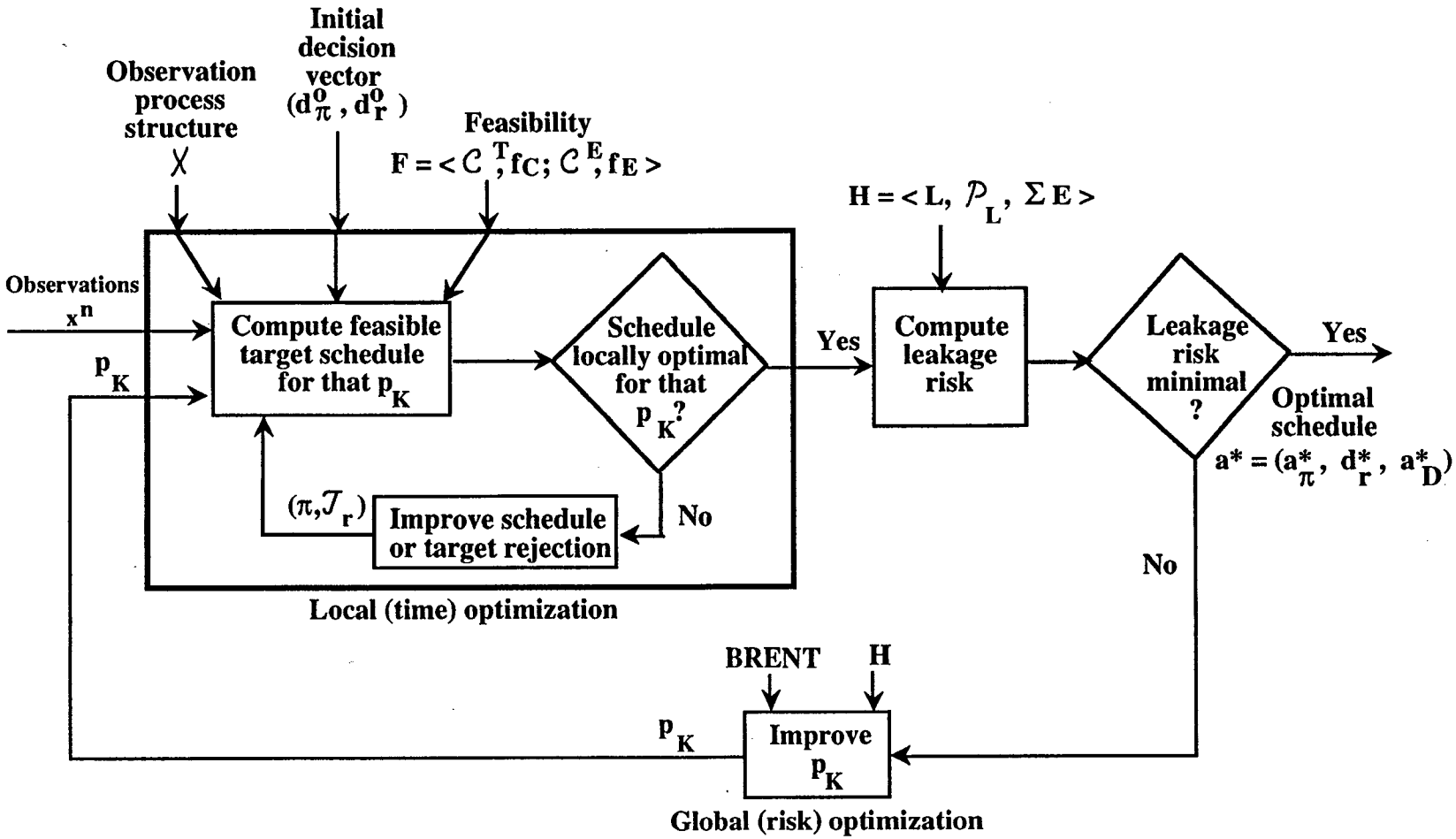

Figure 3.1. The DDTS algorithm consists of two nested optimization loops. Target completion times and target rejection are optimized in the inner loop. In the outer loop, leakage risk is minimized by varying p_K.

Starting with observations x^n, an initial decision vector (d_π^0, d_r^0, d_D^0), a feasibility F and optimization criterion H, the algorithm iterates over the scalar p_K until a minimum value for the total risk $\mathcal{R}$ is obtained. Using the BRENT algorithm, the minimum is reached in very few steps, seldom more than 4 or 5.

In concluding this chapter, we show how the comprehensive threat information available on the platform can be used to transform discrete target scheduling solutions into smooth control hardware commands to reduce the acceleration "jerk" inherent in discrete commands.

3.1 Selecting the Permutation Π

In some situations, the target scheduling problem reduces to the selection of an optimal permutation π^*, and it is then an ordinary target *sequencing* problem. This occurs, for instance, when dwell times t^D are either negligible, or constant and equal. If they are negligible, then any choice of t^D is as good as any other since the completion times $t_{c\pi(i)}$ of Eq. (2.24) are then unaffected by t^D, regardless of the specific value of π or the set $\mathcal{T}_r$ of rejected targets. If dwell times are constant and equal, then their contribution to $t_{c\pi(i)}$ will also be a fixed constant, regardless of π or $\mathcal{T}_r$, because the time function f_C of Eq. (2.29) is then unable to detect any reordering or reassignment of target dwell times. Observe that only in the static case, where distances between targets are fixed, can the equality requirement be removed, and it then suffices that dwell times be constant.

When deadlines are not too "severe" and the leakage probabilities in Eqs. (2.42) and (2.43) are not permutation-dependent, the TSP further reduces to a famous problem known as the Traveling Salesman Problem (Lawler et al. 1985), which we briefly describe below. First, we need some familiar definitions.

A *directed graph* (Roberts 1976) is a pair $G =< V, E >$, where V is a finite set of vertices (the "cities") and $E \subset V \times V$ is a set of pairs of elements from V called the *edges* or *arcs* (also "segments") of G. For any edge $e = (u, v) \in E$, u is called *adjacent to v,* or the *predecessor of v,* and v is called *adjacent from u,* or the *successor of u.* Any edge (u, v) where $u = v$ is called a *loop.*

Consider an (ordered) index set I of size $|I|$. A *path* ("highway") in G is a sequence $S =< v_i : i \in I >$ of vertices, each of which is said to be *visited* by the path. A path is *simple* if no vertex is visited more than once, *closed* if its first element is also its last, a *cycle* if it is simple and closed, and *complete* if it visits every vertex in V. A cycle is also called a sub-tour, and a complete cycle is a *complete tour*, or simply a *tour.* A tour is thus a path which visits every vertex in V once and only once.

To each tour corresponds a permutation $\pi : I \rightarrow I$ of the index set I (see Section 2.2.1.2.1). All tours in a graph can thus be examined in terms of all permutations π on some index set I. Referring to Fig. 3.2, if the original tour is

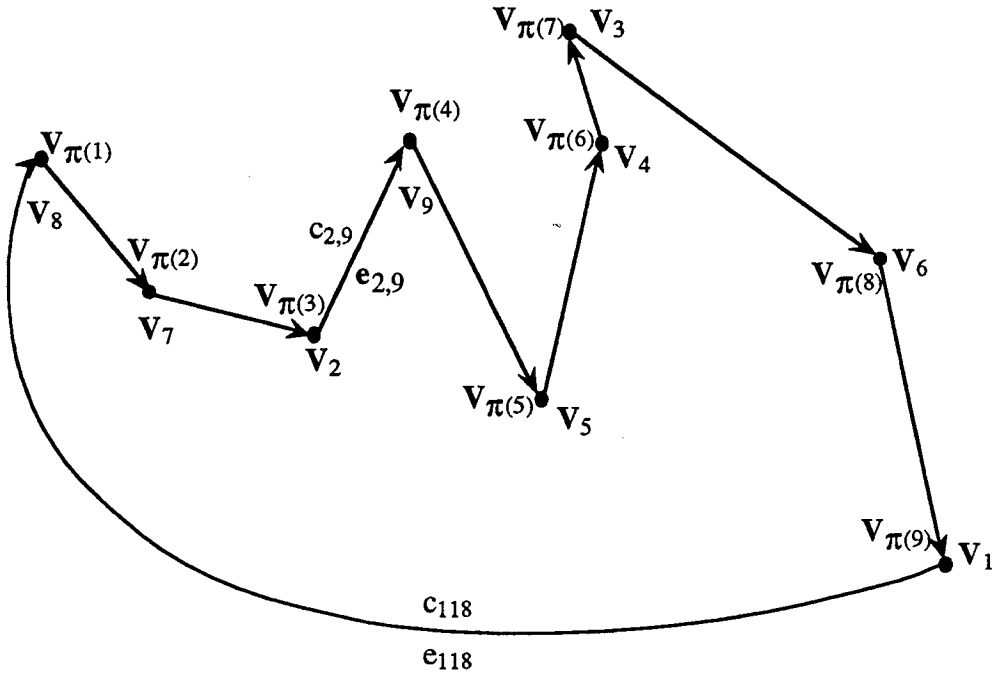

Figure 3.2. An illustration of the original tour τ_0 and its permutation $\tau(\pi)$.

$$\tau_0 =< v_1, \ldots, v_{|I|}, v_1 >, \quad \text{where } |I| = 9,$$

then the tour induced by the permutation π is

$$\tau(\pi) =< v_{\pi(1)}, \ \ldots, \ v_{\pi(1)}, \ \ldots, \ v_{\pi(|I|)}, \ v_{\pi(1)} >,$$

where $v_{\pi(i)}$ is the vertex in position i on the tour $\tau(\pi)$.

Next, consider a cost or distance matrix $[c_{ij}]$ whose entry c_{ij} is the cost of traveling from city v_i to city v_j. The Traveling Salesman Problem is to find a permutation $\pi^* \in \Pi$ such that the tour $\tau(\pi^*)$ minimizes the cost:

$$C(\tau(\pi)) = \sum_{i \in I} c_{\pi(i)\pi(i+1)} \ , \tag{3.1}$$

where $\pi(|I| + 1) = \pi(1)$.

In target scheduling terms, a minimum-risk schedule is found by simply minimizing the sum of all retarget times.

Of particular interest in this report is the more structured situation where the distance or cost matrix $[c_{ij}]$ satisfies the *triangle inequality*,[1]

$$c_{ik} \leq c_{ij} + c_{jk}, \ \{i,j,k\} \subset I \ , \tag{3.2}$$

and the *symmetry condition*,

$$c_{ij} = c_{ji}, \ \{i,j,k\} \subset I \ . \tag{3.3}$$

These relations are obeyed in geometric situations such as those in this report, where cities correspond to points in a metric space, and distances are computed according to the metric of that space. Usually this metric will be the *Euclidean metric* (Loomis and Sternberg 1968), and the problem is then referred to as the Euclidean Traveling Salesman Problem (Lawler et al. 1985).

Next, we enter the realm of heuristics with the admission that the conditions required for the strict applicability of traveling salesman solutions are not frequently satisfied, and that they are often unverifiable. Even move annoying to those seeking accurate solutions is the fact that the traveling salesman problem itself has no efficient solution, since it is NP-hard (Papadimitriou and Steiglitz, 1982).

In spite of these regrettable facts, traveling salesman solutions can provide considerable support because it is generally wise—at least during optimization "start-up"—to minimize the total retarget time, even when dwell times and deadlines are *not* negligible. This policy will usually provide a good starting point for optimization and subsequent corrective steps. Of course, scenarios can be exhibited where the opposite is true since, in some cases, doing nothing for a while will allow the collection of targets to move into a more convenient geometry. We have discussed earlier and shall further discuss below some specific corrective measures that can be taken to minimize the harmful effects associated with such rare cases. Let us now examine a bit more closely the process of constructing an optimal tour.

A traveling salesman problem is typically addressed in three steps:

1. Tour initialization.
2. Vertex selection and insertion.
3. Tour improvement.

[1] We are ignoring here the notable exceptions where certain obstructions or interference may prevent or complicate access to some cities (targets) from some other cities. In such cases, a "shorter" path may be more expensive than a longer one.

3.1.1. Tour Initialization

As in most non-linear optimization problems, the selection of a starting point for tour construction is important. Usually, the starting point is a subtour, a tour on a subset $V' \subset V$ of the vertices. There are many ways of choosing such an initial subtour. In static Euclidean problems, a very successful method is the *Convex Hull Procedure* (Norback and Love 1977; Eddy 1977). Recall that the convex hull of V is the smallest convex set that includes V. Since Flood showed in 1956 that every static Euclidean traveling salesman problem has an optimal solution that visits the cities on the boundary of the hull in the same order as if only the boundary of the hull were traced, this approach is very appealing.

Unfortunately, the construction of even a planar convex hull requires $O(\mid I \mid^2)$ operations (Eddy 1977), and often yields only a small subtour. Hence many vertices may remain to be connected to the subtour to form a tour on *all* of V. An additional difficulty is that vertices—i.e., targets—are added and removed from the tour as old targets are processed and new targets are introduced, and this can considerably slow down the convex hull construction in Cartesian 3-space. As a result of these obstacles, we chose a simpler approach where the initial tour consists of only a single randomly chosen vertex. All of the tour construction work was thus deferred to the vertex selection and insertion process, which we discuss next.

3.1.2. Vertex Selection and Insertion

Once an initial tour has been obtained, the next step is to recursively select vertices currently not on the tour and to insert them in the tour until a full tour is obtained.

There are also several ways to do this. A popular and simple-minded approach is called the Nearest Neighbor Algorithm:

1. Start with an initial subtour consisting of a single vertex v_1.
2. If the current subtour is $S_k =< v_1, \ldots, v_k >, k <\mid I \mid$, select a vertex v_{k+1} not on the subtour, which is nearest to v_k (with respect to $[c_{ij}]$), and add v_{k+1} to the end of the subtour.
3. Stop when a complete tour is obtained.

In the general case of time-varying costs $c_{ij}(t)$, this approach can obviously lead to catastrophic results. But even in the static case, this myopic approach has the obvious drawback that, even though all earlier vertices were at least "locally optimal", the cost $c_{|I|1}$ of the last edge from $v_{|I|1}$ to v_1 may be very high (see Fig. 3.2). In the Euclidean case, of course, the triangle inequality prevents the cost from exceeding the total cost of the previous subtour S_k. More precisely (Reingold et al. 1977), if N_I is the tour cost obtained by the nearest neighbor algorithm, and O_I is the optimal tour cost, then

$$\frac{|\,N_I\,|}{|\,O_I\,|} \leq \frac{1}{2}\Big(\lceil lg(|\,I\,|)\rceil + 1\Big)\,, \tag{3.4}$$

where $\lceil x \rceil$ is the least integer greater than x.

While this is only an upper bound, it is possible to construct cases where the tour cost $|\,N_I\,|$ is *more* than $\frac{1}{3}\,lg\,(|\,I\,|)$ times the cost of an optimal tour. So it is clear that Euclidean structure has been of some help in reducing the worst case error in modeling the static optimal tour cost.

Many variants and extensions of the nearest neighbor approach have been developed and tested (Lawler et al. 1985), and, in spite of their low order of complexity—most are of $O(|\,I\,|^2)$—they have not performed well without additional corrective steps. One algorithm, however, the *farthest insertion algorithm* (FINSA), has almost consistently done better by contradicting the "greedy" policy of nearest neighbor approaches in that the city selected for insertion in the subtours is *farthest* from the subtour. As we shall see, we decompose the permutation selection problem into the selection of "almost-static" subtours, tours whose vertices are visited at almost the same time. We expect that the superiority of FINSA will extend to such tours, and FINSA was thus also our choice for the TSP discussed here. The first step in the FINSA subroutine is the same as for the nearest neighbor algorithm. But in Step 2, it chooses the city or vector v_k not on the subtour which *maximizes* the distance $\min\,\{c_{jk} : v_j \in S_k\}$ from v_k to the current subtour S_k. Once k is found, the greedy policy is resumed, and v_k is inserted as the (immediate) successor to v_i and the (immediate) predecessor to v_j if, and only if, v_i precedes v_j in the current subtour and the *local insertion cost* $c_{ik} + c_{kj} - c_{ij}$ is a minimum.

3.1.3. Tour Improvement

In spite of impressive results in many applications, the best that appears known about the accuracy of FINSA is that it generates static tours which approach 3/2 times the optimal tour length. Because farthest insertion does not correspond in any direct way to any algorithm for generating minimum spanning trees, it has been more difficult to analyze than other insertion algorithms, and new techniques will be required to derive tight upper bounds. In many applications including the TSP, the error uncertainty associated with FINSA is not acceptable and *tour improvements* must be made.

Most reliable tour improvement procedures are *edge exchange procedures* (Lawler et al. 1985) where r edges in a tour are exchanged for r edges not currently on the tour as long as the exchange still yields a tour and the length of the exchanged tour is less than the length of the previous tour. Such edge-exchange procedures are referred to as *r-optimal* procedures (Lin and Kernighan 1973), where r is the quantity of edges exchanged at each iteration. A tour is called *r-optimal* if there is no further exchange of r edges which can result in a shorter tour.

Unfortunately, the complexity of r-opt procedures increases rapidly as r is increased, not only because all r-exchanges must be tested—potentially an $O(r!)$ process—but because every exchange changes the tour, and a new $O(r!)$ process may have to be initiated at every exchange. That is essentially why r-opt methods have escaped rigorous complexity analysis and no tight complexity or accuracy bounds are known. In spite of this, r-opt procedures have been very successful, even when r only equals 2, and this is the method we have chosen for the TSP.

To gain some technical insight into the 2-opt procedure, refer to the simple path of Fig. 3.3 where just one pairwise exchange is shown. The only way to obtain a simple path after removing edges e_1 and e_5, without altering the terminal nodes T_o and T_{j+1}, is to insert two new edges, e_6 and e_{10}. Observe that the new path segment (e_6, e_7, e_8, e_9, e_{10}) from T_a to T_f (shown as a dashed line) involves a reversal from T_e to T_b. This may be very important in time-varying networks where a node such as T_e is reached too late and nodes like T_b are reached early. The skipping of nodes T_b, T_c, and T_d on the way to T_e may obviously gain considerable time and may avoid the tardiness of node T_e. While the furthest insertion approach (FINSA) is excellent in finding shortest paths through topologically static networks, considerable improvements may be needed when nodes of the network are moving and when the processing time at each node is position-and-time-dependent, and the 2-opt improvement procedure provides an excellent balance between performance and speed.

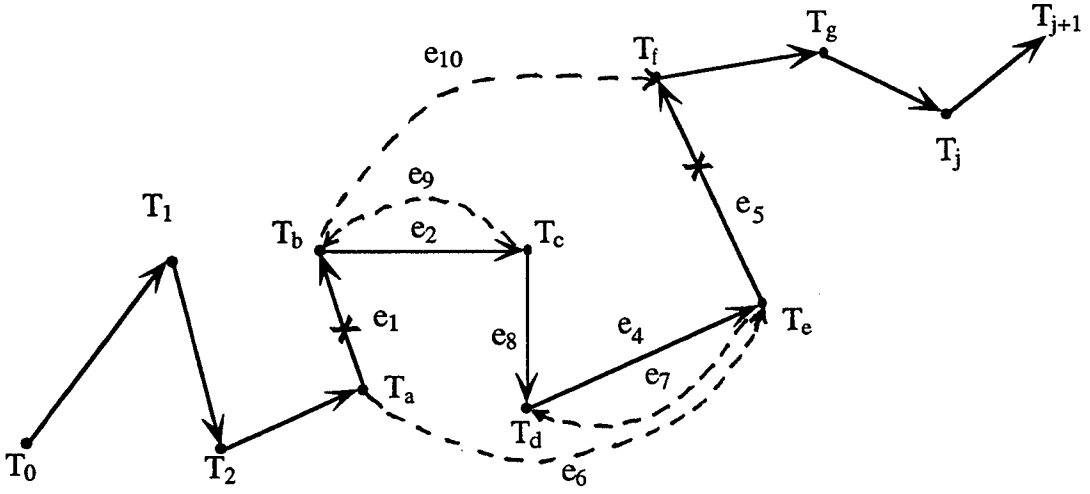

Figure 3.3. Changing edges e_1 and e_5 for edges e_6 and e_{10} in the 2-opt tour improvement procedure. The new path segment is shown as a dashed line.

3.2 Target Rejection

Even when schedules are optimal, some deadlines may not be met, as we discussed in Section 2.2.2. In such cases, and depending upon the penalties associated with target tardiness, one or more "unimportant" targets may have to be rejected so that "important" targets can be processed by their deadline. The problem of developing job rejection criteria is not new, and the work presented in this chapter reflects considerable work done by many others on similar—but much simpler—problems. Even for simple problems, proofs of optimality are scarce, and our approach is necessarily heuristic.

As before, we have attempted to use previous results as limiting cases of our problem, hoping that the special way in which we have combined problem parameters has produced a composite rejection criterion that is near-optimal. First, we derive the criterion in general terms. Then we define each of its constituent elements in rigorous detail.

3.2.1. The Rejection Criterion r

Target rejection can be discussed in terms of three simple but competing principles:

1. Ignore targets with large processing times.
2. Ignore targets whose leakage risk is low.
3. Ignore targets whose deadlines are soft.

The challenge, of course, is to combine these conflicting criteria into a *rational* composite that resolves the conflicts while approaching optimality. Starting with the temporal aspects, it was shown in Conway et al. (1967) that scheduling jobs in accordance with the SPT schedule (shortest processing time first) minimizes the *mean flow time* $\overline{F}$, the mean time elapsed from job release to job completion. In French (1982), $\overline{F}$ is also shown to be equivalent to *mean completion time* $\overline{C}$. It was also shown in Smith (1956) and Dyer and Wolsey (1990) that, when jobs have different weights or values, the weighted sum of completion times is minimized if jobs are scheduled in accordance with the ratios of their processing time to their weight, smallest ratios first. Of particular relevance to the TSP is the machine scheduling problem of minimizing the number of tardy jobs, those whose completion time exceeds their deadline. Extending the work of Moore (1968) to the case where consistent release (availability, ready) times are considered, an optimal solution was derived in Kise et al. (1978) with a rule that rejects jobs which marginally contribute the most to completion times. This rule reduces to the SPT schedule when release constraints are ignored. While none of these objective functions exactly match our own, they—together with more complicated numerical rules (Abdul-Razaq 1990)—strongly suggest that our heuristic should contain target processing time τ and target values or weights w in the form of a ratio $r = w/\tau$, larger ratios indicating greater importance.

In target scheduling, processing time is defined in terms of the dwell time t^D and the retarget time t^R, and we are interested in the marginal time gained by ignoring a given target. This time gain is simply $\tau = t^{D^*} + \Delta t^R$, where t^{D^*} is the dwell time *conditional* to the target type θ^* assumed by the battle or platform manager, and Δt^R is the retarget time earned by skipping the target. Observe that in the TSP, these processing times are *controllable* in the sense that they are adjusted on the basis of previous and future decisions. Recently, some renewed attention has been devoted to problems with controlled processing times (Daniels 1990; Vickson 1980a,b), and the simple rule outlined here agree with the results of that work.

In our risk approach to scheduling, we admit that target values or priorities depend on many statistical factors, and we use *expected leakage loss* as a measure of target value. This loss is simply $p_M \cdot V$, where V is the value of the target, and p_M is the probability of miss—i.e., the probability that the target will leak through if its processing is attempted. But p_M is determined not only by weapon lethality and target vulnerability; it is also determined by the ability of the weapons platform to select and identify the target. Thus, if p_M is the probability that the target is missed, and if we assume that every misidentified target leaks through,

$$\begin{aligned} p_M &= \left(p_M \mid_{ID}\right) p_{ID} + \left(p_M \mid_{\overline{ID}}\right)\left(1 - p_{ID}\right) \\ &= \left(p_M \mid_{ID}\right) p_{ID} + \left(1 - p_{ID}\right), \end{aligned} \tag{3.5}$$

where

p_{ID} is the probability of correctly identifying the target, and

$p_M \mid_{ID}$ $\left(p_M \mid_{\overline{ID}}\right)$ is the probability that the target is missed if it has been correctly (incorrectly) identified.

Expressed in terms of target destruction, we get

$$\begin{aligned} p_K = 1 - p_M &= \left(1 - p_L \mid_{ID}\right) p_{ID} \\ &= \left(p_K \mid_{ID}\right) p_{ID}, \end{aligned} \tag{3.6}$$

where $p_K \mid_{ID}$ is the probability that the target is destroyed given that it has been correctly identified.

As we discussed in Section 2.2.2.2, our approach allocates to each target an equal value of $p_K \mid_{ID}$. Carrying along $p_K \mid_{ID}$ is thus superfluous, and we use only p^*_{ID}, the probability of correctly identifying the target, given that the target was classified as being of type θ^*.

The target loss value L itself is uncertain, of course, since it depends on target type θ, which is a random variable. To exploit all the available target type information and to

reflect the classification rules embedded in the type-discrimination algorithm, we use the Bayes *posterior expected value* V^* of the target value V.

Finally, the deadline hardness $\alpha \in [0,1]$ is only used as a multiplier in the rejection ratio. Combining all these arguments, we obtain the composite target rejection criterion

$$\boxed{r = \frac{\alpha \cdot p^*_{ID} \cdot V^*}{t^{D*} + \Delta t^R}} \tag{3.7}$$

where

α is the deadline hardness.

p^*_{ID} is the probability of correctly identifying the target type, conditional to deciding its type is θ^*.

V^* is the expected target loss conditional to θ^*.

t^{D^*} is the target dwell time conditional to θ^*.

Δt^R is the retarget time earned by skipping the target.

While the obvious strategy is thus to skip a target whose ratio is *least*, note the interesting circumstance where such a least target precedes an "early" target, one where waiting occurs. Recall that the processing of a target cannot be started before its release time. Hence there is no point in skipping a target that precedes an early target since that will simply increase the waiting time at the early target. Before we address this subtlety, however, let us define the rejection ratio in more rigorous terms.

3.2.1.1. The Dwell and Retarget Times Δt^{D^*} and Δt^R. The total time gained by skipping a target is $\Delta t = t^{D^*} + \Delta t^R$, the sum of the target dwell time and the retarget time gained. Recall that the dwell time t^{D^*} is conditional to the selected target type θ^*. While the entire dwell time is gained when a target is skipped, the retarget time gained depends on release times and completion times of neighboring targets. Referring to Fig. 3.4, where the skipping of target T_j is considered, let t^C_x be the completion time of T_x, $T^R_{x,y}$ the retarget time from T_x to T_y, t^A_x the availability time of T_x, and $t^{D^*}_y$ the dwell time of T_y. Then

$$t^C_j = \max\left\{t^C_{j-1} + t^R_{j-1,j},\ t^A_j\right\} + t^{D^*}_j \ . \tag{3.8}$$

$$t^C_{j+1} = \max\left\{t^C_j + t^R_{j,j+1},\ t^A_{j+1}\right\} + t^{D^*}_{j+1} \ . \tag{3.9}$$

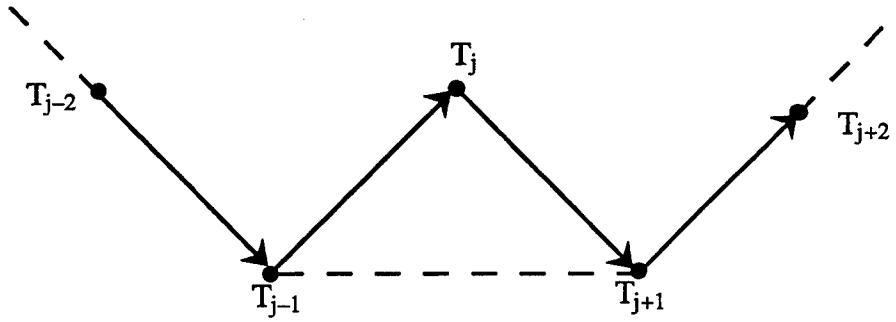

Figure 3.4. Target T_j is skipped by traveling directly from T_{j-1} to T_{j+1}.

The total gain when skipping T_j is thus

$$\begin{aligned}
\Delta t_j &= t^C_{j+1} - \max\left\{t^C_{j-1} + t^R_{j-1,j+1},\ t^A_{j+1}\right\} + t^{D^*}_{j+1} \\
&= \max\left\{t^C_j + t^R_{j,j+1},\ t^A_{j+1}\right\} + t^{D^*}_{j+1} - \max\left\{t^C_{j-1} + t^R_{j-1,j+1},\ t^A_{j+1}\right\} - t^{D^*}_{j+1} \\
&= \max\left\{t^C_j + t^R_{j,j+1},\ t^A_{j+1}\right\} - \max\left\{t^C_{j-1} + t^R_{j-1,j+1},\ t^A_{j+1}\right\} \\
&= \max\left\{\max\left\{t^C_{j-1} + t^R_{j,j+1},\ t^A_j\right\} + t^{D^*}_j + t^R_{j,j+1},\ t^A_{j+1}\right\} \\
&\qquad - \max\left\{(t^C_{j-1} + t^R_{j-1,j+1},\ t^A_{j+1}\right\}. \qquad (3.10)
\end{aligned}$$

Observe that, when release times (t^A_x) are soft, the total retarget and dwell time gained is simply

$$\Delta t_j = t^R_{j-1,j} + t^{D^*}_j + t^R_{j,j+1} - t^R_{j-1,j+1}\,. \qquad (3.11)$$

3.2.1.2. The Deadline Hardness Constant α. Each target T_j is assigned a possibly different constant $\alpha_j \in [0, 1]$.

3.2.1.3. The Identification Probability p^*_{ID}. We compute the probability p^*_{ID} of correct identification using a Bayesian approach,* where p^*_{ID} is a *posterior* estimate derived from *prior* estimates $\pi(\theta_u) = \text{prob}\ (T \text{ is of type } \theta_u)$ and additional information about intervening decisions. More precisely, let

p^*_{ID} = prob (correctly deciding target type $|_{\text{decide type is } \theta^*}$).
$p(\theta_u\ |_{\theta_v})$ = prob (decide type is $\theta_u\ |_{\text{type is } \theta_v}$).
$q(\theta_v\ |_{\theta_u})$ = prob (type is $\theta_v\ |_{\text{type is } \theta_u}$).

* Ferguson (1967); Berger (1980); Duda and Hart (1973); Jain (1989).

Then, using Bayes' rule,

$$p^*_{ID} = q(\theta^* \mid_{\theta^*}) = \frac{\pi(\theta^*)p(\theta^* \mid_{\theta^*})}{\sum_{v=1}^{M} \pi(\theta_v)p(\theta^* \mid_{\theta_v})} , \tag{3.12}$$

where M is the quantity of possible target types.

3.2.1.4. The Posterior Loss L*. When a target of type θ_u leaks through, a loss $L(\theta_u, a) = V_u$ is experienced. Given that target types are uncertain, Bayes' theorem is also used here to compute the posterior loss $L^* = E(L(\theta_u(\omega), a) \mid_{\theta^*})$ of the target leaking through, conditional to deciding that its type is θ^*. Let T_{θ_u} denote a target of type θ_u, and let

$p^*_M(\theta_u)$ = prob (missing target $T_{\theta_u} \mid_{\text{decide type is } \theta^*}$).

$p^*_M(T_\theta, \theta_u)$ = prob (missing target T_θ *and* target is of type $\theta_u \mid_{\text{decide type is } \theta^*}$).

$p^*_K(\theta_u)$ = prob (destroying target $T_{\theta_u} \mid_{\text{decide type is } \theta^*}$).

$p^*_K(\theta_u \mid_{ID})$ = prob (destroying $T_{\theta_u} \mid_{\text{decide } \theta^* \text{ and correctly identify } T_{\theta_u}}$).

$p^*_K(\theta_u \mid_{\overline{ID}})$ = prob (destroying $T_{\theta_u} \mid_{\text{decide } \theta^* \text{ and incorrectly identify } T_{\theta_u}}$).

Then

$$\begin{aligned} p^*_K(\theta_u) &= p^*_K(\theta_u \mid_{ID}) p^*_{ID} + \left(p^*_K(\theta_u) \mid_{\overline{ID}} \right)(1 - p^*_{ID}) \\ &= \left(p^*_K(\theta_u) \mid_{ID} \right) p^*_{ID}(\theta_u) , \end{aligned} \tag{3.13}$$

where we assume that any incorrectly identified target leaks through.

The expected leakage loss associated with a target, *posterior to* deciding that its type is θ^*, is thus

$$\begin{aligned} L^* = E(L(\theta(\omega), a) \mid_{\theta^*}) &= \sum_{u=1}^{M} V_u p^*_M(T_u, \theta_u) \\ &= \sum_{u=1}^{M} V_u q(\theta_u \mid_{\theta^*}) p^*_M(\theta_u) . \end{aligned} \tag{3.14}$$

But

$$q(\theta_u \mid_{\theta^*}) = \frac{\pi(\theta_u)p(\theta^* \mid_{\theta_u})}{\sum_{v=1}^{M} \pi(\theta_v)p(\theta^* \mid_{\theta_v})} ,$$

and

$$
\begin{aligned}
p_M^*(\theta_u) &= 1 - p_K^*(\theta_u) = 1 - \Big(p_K^*(\theta_u) \mid_{ID} \Big) p_{ID}^*(\theta_u) \\
&= 1 - p_K p_{ID}^*(\theta_u) \\
&= \begin{cases} 1 & , \quad \theta_u \neq \theta^* \\ (1 - p_K) & , \quad \theta_u = \theta^* \, , \end{cases}
\end{aligned} \tag{3.15}
$$

where p_K is a constant independent of type θ_u, reflecting the fact that each target is assigned an equal $p_K^*(\theta_u) \mid_{ID}$. Concluding,

$$
\begin{aligned}
L^* &= E\Big(L(\theta(\omega),\ a) \mid_{\theta^*} \Big) = \sum_{\substack{u=1 \\ \theta^* \neq \theta_u}}^{M} \frac{V_u \pi(\theta_u) p(\theta^* \mid_{\theta_u})}{\sum_{v=1}^{M} \pi(\theta_v) p(\theta^* \mid_{\theta_v}} + V_{\theta^*}(1 - p_K) p_{ID}^* \\
&= \sum_{u=1}^{M} \frac{V_u \pi(\theta_u) p(\theta^* \mid_{\theta_u})}{\sum_{v=1}^{M} \pi(\theta_v) p(\theta^* \mid_{\theta_v})} - V_{\theta^*} p_{ID}^* p_K \, .
\end{aligned} \tag{3.16}
$$

3.3 Selecting the Swell Time Vector $\mathbf{t^D}$

In the TSP, increasing dwell (processing) time of a target decreases its leakage probability. But it also delays the completion time of subsequent targets, and may result in missed deadlines. In most scenarios, in fact, dwell time is the major contributor to target completion time, and tradeoff between dwell time and tardiness must therefore be resolved. A strict minimization of expected loss, however, could produce different dwell times for different targets and would typically require an $|I|$-dimensional nonlinear optimization. During the boost phase, one way around this computational burden is to exploit the fact that the dwell time required to achieve a given destruction probability p_K is very sensitive to p_K, increasing rapidly as p_K exceeds a nominal value p_{NOM} (typically near 0.8) (see Fig. 3.5). When one target is more valuable than another, the natural temptation to increase its p_K is quickly damped when small increases in p_K cause large increases in t^D and disproportionate losses due to tardiness are incurred, as shown in Figs. 3.5 and 3.6 for one typical scenario.

As we discussed in Section 2.2.2.2, kill probabilities tend to bunch around a nominal value, and the multidimensional optimization may be replaced by a scalar minimization by assigning to each target a fixed p_K, and by varying this p_K until a minimum risk is obtained. The individual target dwell times can then be computed from this single p_K^* via an inversion process, and that is what we have done. An analogous situation prevails during the midcourse discrimination phase where, instead of p_K, the optimization variable is the *particle return count* N_D. Both inversion processes involve several

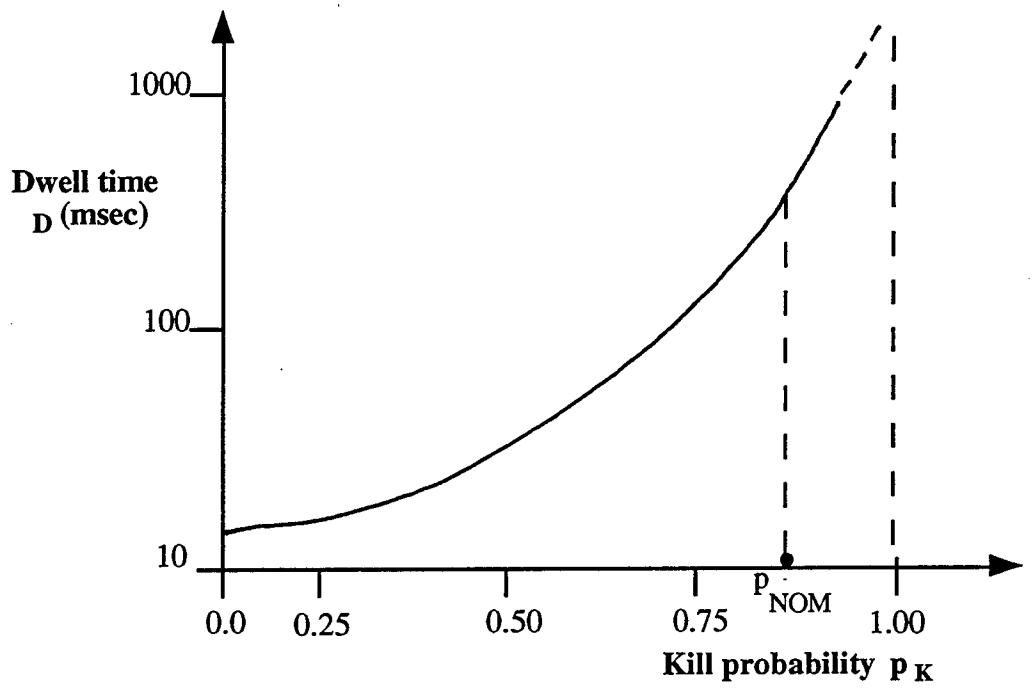

Figure 3.5. Dwell time is very sensitive to p_K above p_{NOM}.

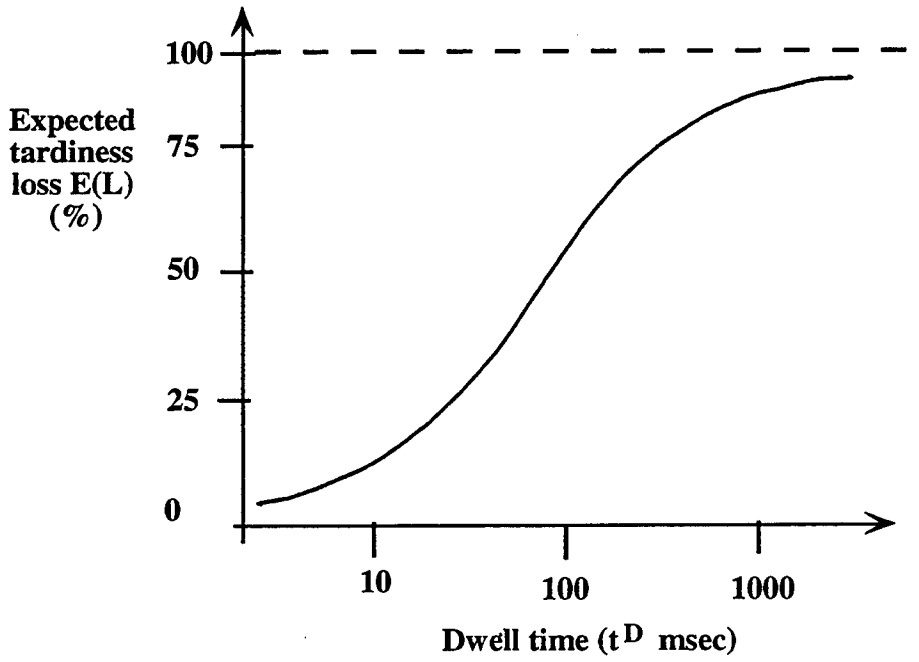

Figure 3.6. Tardiness losses increase rapidly as dwell times are increased.

subtle mathematical arguments peculiar to the defense phase under consideration, and no generalization would have been productive. Their derivation was deferred to future sections where each phase is discussed separately.

A legitimate concern, of course, is that the minimum risk may only be a *local* and not a *global* minimum. Individual target variations may in fact produce as many local minima as there are targets, as we explained in Section 2.2.2.2. But the bunching of dwell time values also discussed earlier essentially limits the effects of these variations to a ripple dominated by a unimodal trend, as shown in Fig. 3.7. To gain control over these ripples, we modified Brent's Method (Brent 1973; Press et al. 1988) to allow a tolerance setting sufficiently large that errors of magnitude less than ϵ can be tolerated. The value of the dwell time t^D derived from p_K^* is then also within acceptable error bounds.

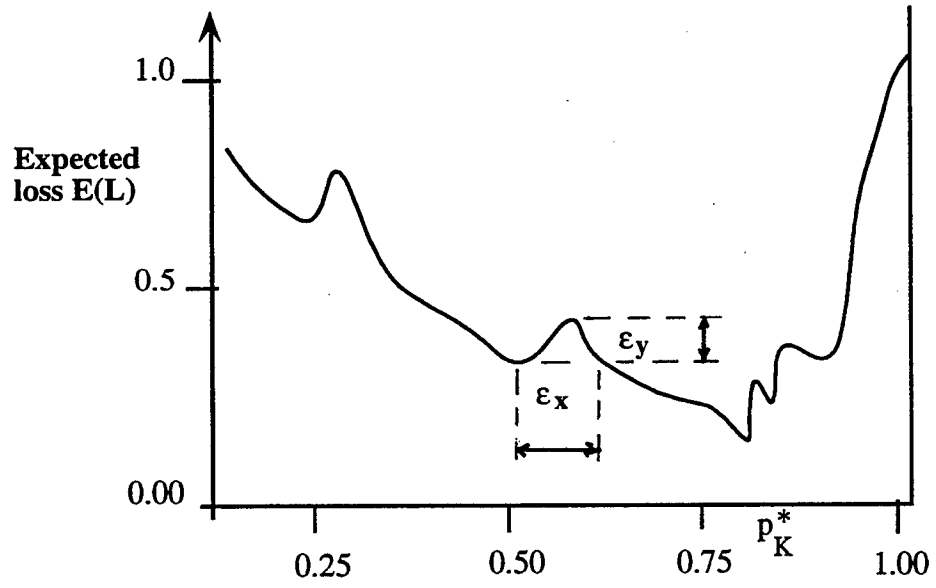

Figure 3.7. Local minima are strongly dominated by the p_K global minimum, and ripples of size less than ϵ have negligible affect on the optimum value p_K^*.

3.3.1. Deriving the Optimal Dwell Time t^{D*} from p_K^*

In this section we continue the boost phase emphasis to find the dwell time t^{D^*} required to achieve a given p_K.

To appreciate the importance of geometry in this derivation, consider Fig. 3.8 where we show the target's vulnerable set A_{AIM} with radius r_{AIM}, and the DEW's energy deposition set A_{DEW}, with radius r_{DEW}. Target destruction is determined by the

DEW energy deposited in the critical intersection $A_{EFF} = A_{AIM} \cap A_{DEW}$ called the *effective set*, whose area we define as the *effective area* $|A_{EFF}|$. Clearly, no damage is done to T if $|A_{EFF}| = 0$, and maximum damage is done when maximum energy or power is delivered to A_{AIM}, which occurs when $|A_{EFF}| = min\{|A_{AIM}|, |A_{DEW}|\}$ if we assume a uniform energy density flow through A_{DEW}. If another pulse profile (e.g., Gaussian, "tophat") is valid, a similar argument may be used with appropriate energy normalization. Hence A_{EFF} is a major factor in the lethality L of the DEW, as shown in the following definition.

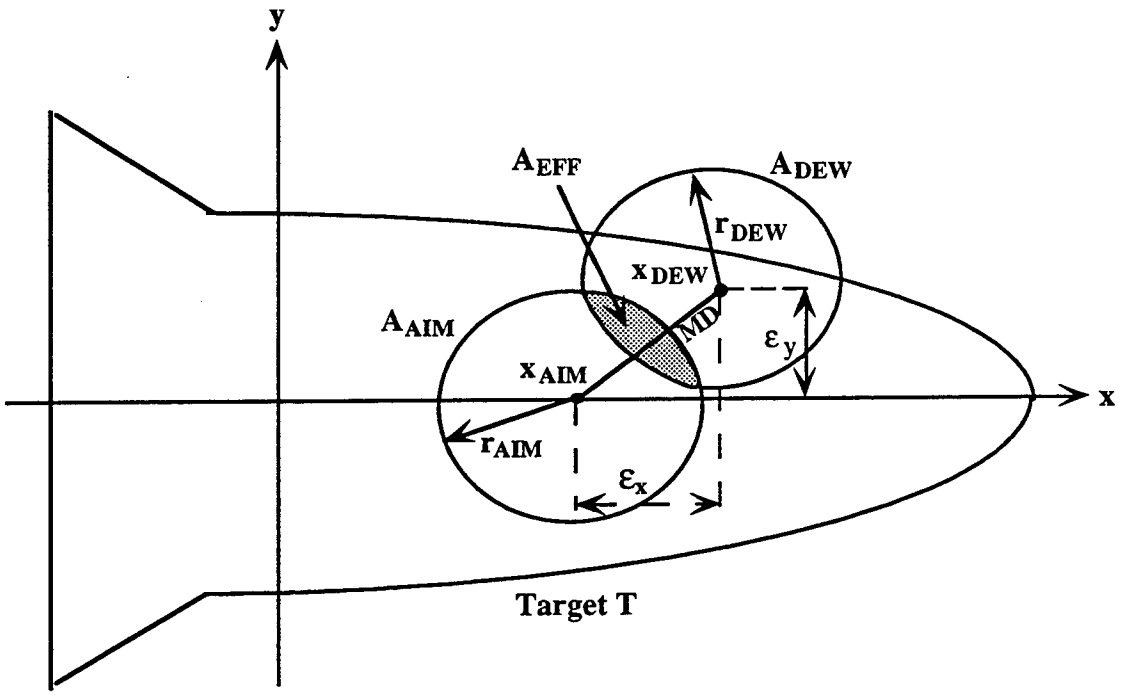

Figure 3.8. Geometry used to compute p_K and t^D.

Definition

We define DEW lethality as

$$L = \frac{Qt^D\,|A_{EFF}|}{A^o_{EFF}} \left(joules\,|_{cm^2}\right) \tag{3.17}$$

where

$$Q = \frac{\pi P D^2 cos\alpha}{4\lambda^2 R^2 exp\left(\frac{R^2}{k_T cos\,\alpha}\right)} \left(watts\,|_{cm^2}\right), \text{ and} \tag{3.18}$$

D is the diameter of the DEW optics (m).
α is the aspect angle between the DEW beam and the target.
λ is the DEW wavelength (m).
R is the distance from the DEW to the target (m).
P is the DEW power ($w\mid_{\text{steradian}}$).
t^D is the DEW dwell time.

$A^0_{EFF} = \min\left\{ \mid A_{AIM} \mid, \mid A_{DEW} \mid \right\}$.

The "miss distance" $MD = \mid X_{AIM} - X_{DEW} \mid$ is due to random error ϵ_x and ϵ_y caused by random angular errors E_{θ_x} and E_{θ_y} due to uncertainties in acquisition, tracking, pointing (ATP), and also in sensing and aimpoint location. A_{EFF} is thus a random variable, and so is the lethality L. The probability that a target leaks through may thus be expressed as follows:

$$p_L = p(L < H)\,, \tag{3.19}$$

where H is the (random) hardness of the target. Both the distributions of L and H are therefore required to obtain p_L. In practice, the hardness H is assumed Gaussian with mean μ_H and standard deviation σ_H. But the derivation of the distribution of L is more difficult. We proceed in steps. First, we derive the functional relationship relating L to the underlying random angular ATP and aimpoint location errors E_{θ_x} and E_{θ_y}. Then we derive the distributions of the total angular error E_θ, the effective area $\mid A_{DEW} \mid$, and the lethality L. We conclude with the derivation of dwell time t^{D*} from the leakage probability p_L.

3.3.1.1. The Functional Relationship Between L and E_θ. Due to assumed circular symmetry, the miss distance is $MD = RE_\theta$,

where

$E_\theta = \left(E^2_{\theta_x} + E^2_{\theta_y}\right)^{1/2}$, the *circular error*,
$E_{\theta_x} = b_{\theta_x} + \epsilon_{\theta_x}$, the angular error along x,
$E_{\theta_y} = b_{\theta_y} + \epsilon_{\theta_y}$, the angular error along y,
b_{θ_x} and b_{θ_y} are constant bias terms, and
ϵ_{θ_x} and ϵ_{θ_y} are random error terms.

To obtain the effective area $\mid A_{EFF} \mid = \mid A_{AIM} \cap A_{DEW} \mid$ as a function of the miss distance $MD = RE_\theta$, consider Fig. 3.9, where the case $\mid A_{DEW} \mid \geq \mid A_{AIM} \mid$ is illustrated. If we define $A^0_{EFF} = min\left\{ \mid X_{AIM} \mid, \mid X_{DEW} \mid \right\}$, and if $A_{AIM} \subset A_{DEW}$, then $\mid A_{EFF} \mid = \mid A_{AIM} \mid = A^0_{EFF}$. At the other extreme, if $A_{AIM} \cap A_{DEW} = \phi$, then $\mid A_{EFF} \mid = 0$. Between these two extremes, we assume that $\mid A_{EFF} \mid$ decreases linearly. The case $\mid A_{DEW} \mid \geq \mid A_{AIM} \mid$ is similar, and we obtain the function shown in Fig. 3.10.

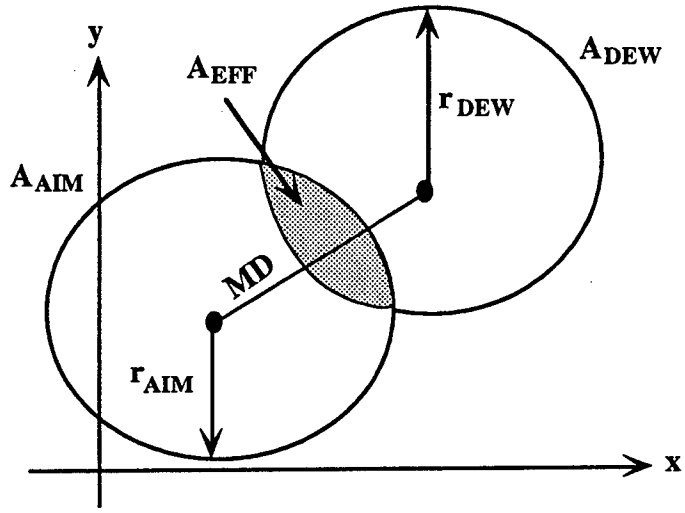

Figure 3.9. Geometry used to compute A_{EFF} $(MD = RE_\theta)$.

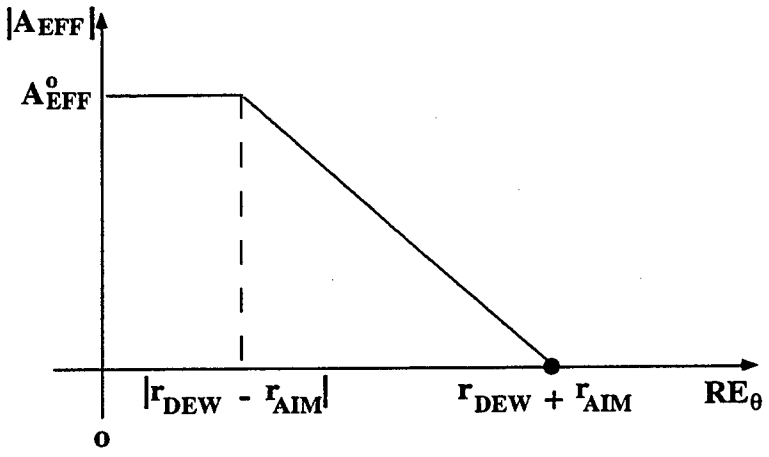

Figure 3.10. The function $|A_{EFF}|$ (RE_θ).

More explicitly,

$$| A_{EFF} |= \begin{cases} A^0_{EFF} & , \quad RE_\theta < | r_{DEW} - r_{AIM} | \\ 0 & , \quad RE_\theta > r_{DEW} + r_{AIM} \\ A^0_{EFF}\Big(1 - \dfrac{RE_\theta - | r_{DEW} - r_{AIM} |}{2min\{r_{DEW}, r_{AIM}\}}\Big) & , \quad \text{otherwise} . \end{cases} \tag{3.20}$$

This function exhibits $| A_{EFF} |$ as a function of the random miss distance $MD = RE_\theta$, and of three additional parameters A^0_{EFF}, r_{DEW} and r_{AIM}. Combining Eq. (3.20) with Eq. (3.16), we obtain lethality L as a function of RE_θ, as desired. To get the distribution of L we first compute the distribution of MD.

3.3.1.2. The Probability Distribution of MD. Recall the angular errors $E_{\theta_1} = b_{\theta_1} + \epsilon_{\theta_1}$ and $E_{\theta_2} = b_{\theta_2} + \epsilon_{\theta_2}$, whose distributions are $N(b_{\theta_1}, \sigma_1)$ and $N(b_{\theta_2}\ \sigma_2)$, respectively. Therefore, $RE_{\theta_1} \sim N(Rb_{\theta_1}\ R\sigma_1)$ and $RE_{\theta_2} \sim N(Rb_{\theta_2}, R_{\sigma_2})$. We seek the distribution of $RE_\theta = (E^2_{\theta_1} + E^2_{\theta_2})^{1/2}$. To simplify notation, consider two independent random variables X and Y with means μ_x and μ_y and common variance σ^2_0. Their joint density is

$$f_{XY}(x,y) = \frac{1}{(2\pi\sigma^2_0)^{1/2}}\, e^{-\Big[\Big((x-\mu_x)^2 + (y-\mu_x)^2\Big)/2\sigma^2_0\Big]} . \tag{3.21}$$

If we define $\mu = (\mu^2_x + \mu^2_y)^{1/2}$, the probability density of the random variable $Z = (X^2 + Y^2)^{1/2}$, as given by Papoulis (1965), is

$$f_Z(z) = \begin{cases} \dfrac{z}{\sigma^2_0}\, exp\Big(-(z^2+\mu^2)/2\sigma^2_0\Big) I_0\Big(\dfrac{z\mu}{\sigma^2_0}\Big) & , \quad z > 0 \\ 0 & , \quad z < 0 \end{cases} \tag{3.22}$$

where I_0 is the modified Bessel function of order zero:

$$I_0(v) = \frac{1}{2\pi}\int_0^{2\pi} e^{v\cos\,\theta} d\theta = \sum_{n=0}^{\infty} \frac{v^{2n}}{2^{2n}(n!)^2} . \tag{3.23}$$

Rewriting this, and assuming $z \geq 0$,

$$f_Z(z) = \frac{z}{\sigma^2_0} exp\Big(-\frac{z^2}{2\sigma^2_0}\Big) exp\Big[-\frac{\mu^2}{2\sigma^2_0}\Big] \sum_{n=0}^{\infty} \frac{1}{(n!)^2}\Big(\frac{\mu z}{2\sigma^2_0}\Big)^{2n} , \tag{3.24}$$

and the complement of the cumulative distribution of Z is

$$p(Z > z) = exp\Big(-\frac{\mu^2}{2\sigma_0^2}\Big)\int_z^\infty \frac{v}{\sigma_0^2} exp\Big[-\frac{v^2}{2\sigma_0^2}\Big]\sum_{n=0}^{\infty}\frac{1}{(n!)^2}\Big(\frac{\mu v}{2\sigma_0^2}\Big)^{2n} dv\ .$$

In Appendix B we show that this distribution may be approximated by a normal distribution:

$$p(Z > z) = \Phi\Bigg(\frac{\sigma_0 - z^2 + \mu^2}{\Big[2(z^2+\mu^2)\Big]^{1/2}}\Bigg)\ . \tag{3.25}$$

For our problem, $Z = RE_\theta$ and $\mu = \Big((Rb_{\theta_1})^2 + (Rb_{\theta_2})^2\Big)^{1/2}$ and $\sigma_0 = R\sigma$, hence:

$$p(RE_\theta > d) = \Phi\Big(\frac{(R\sigma)^2 - d^2 + \mu^2}{(2R^2\sigma^2(d^2+\mu^2))^{1/2}}\Big),\ \ d \geq 0. \tag{3.26}$$

3.3.1.3. The Probability Distribution of L. We start by deriving the cumulative distribution (CDF) of $|A_{EFF}|$. Referring to Fig. 3.8 and Eq. (3.20), define

$$p_0 = prob\Big(\ |A_{EFF}| = 0\Big) = prob\Big(RE_\theta > r_{DEW} + r_{AIM}\Big)\ ,$$

and

$$p_1 = prob\Big(\ |A_{EFF}| = A^0_{EFF}\Big) = prob\Big(RE_\theta < |\ r_{DEW} - r_{AIM}\ |\ \Big)\ .$$

We approximate the CDF of $|\ A_{EFF}\ |$ as follows (see Fig. 3.11):

$$p(|A_{EFF}| \leq a) = \begin{cases} 0 & ,\ a < 0 \\ p_0 & ,\ a = 0 \\ 1 - p_1 & ,\ a = A^0_{EFF} \\ 1 & ,\ a > A^0_{EFF} \\ p_0 + \frac{(1-p_1-p_0)a}{A^0_{EFF}} & ,\ 0 < a < A^0_{EFF} \end{cases} \tag{3.27}$$

where the events $\{|\ A_{EFF}\ | < 0\}$ and $\{|\ A_{EFF}\ | = A^0_{EFF}\}$ are equivalent to the events $\{RE_\theta > r_{DEW} + r_{AIM}\}$ and $\{RE_\theta < |\ r_{DEW} - r_{AIM}\ |\}$, respectively.

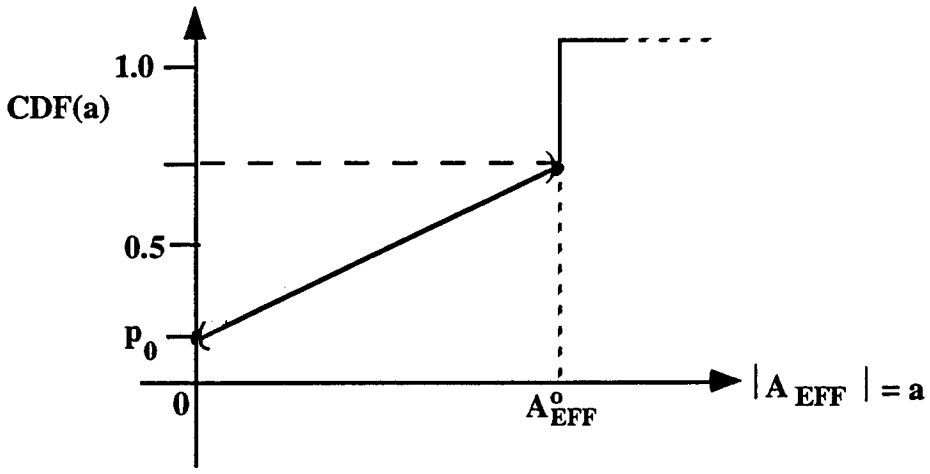

Figure 3.11. The Cumulative Distribution Function (CDF) of $|A_{EFF}|$.

Finding the CDF of L is now easy:

$$p(L \le l) = \begin{cases} 0 & , \ l < 0 \\ p_0 & , \ l = 0 \\ l - p_1 & , \ l = Qt^D \\ 1 & , \ l > Qt^D \\ p_0 + \frac{(1 - p_1 - p_0)}{A^0_{EFF}} & , \ 0 < l < Qt^D \end{cases} \tag{3.28}$$

3.3.1.4. Deriving the Dwell Time t^{D^*}. Recall that $p_K = p(L > H)$. But

$$\begin{aligned} p_K &= \int_H p\Big((L > h) \wedge (H = h)\Big) dh \\ &= \int_H p\Big((L > h) \mid_{H=h}\Big) f_H(h) dh \ . \end{aligned}$$

Using the approximation $f_H(h)$ to the Gaussian density shown in Fig.3.12, we have

$$p_K = \frac{1}{3\sigma_H} \int_{\mu_H - 1.5\sigma_H}^{\mu_H + 1.5\sigma_H} p(L > h) dh \ . \tag{3.29}$$

Substituting the expression for L,

$$p_K(t^D) = \frac{1}{3\sigma_H} \int_{\mu_H - 1.5\sigma_H}^{\mu_H + 1.5\sigma_H} p\left(A_{EFF} \ge \frac{h A^0_{EFF}}{Qt^D}\right) dh \ . \tag{3.30}$$

Even though we have the distribution of A_{EFF} (see Eq. 3.27), and p_K is monotonically increasing in t^D, the derivation of the unique inverse $t^{D^*} = p_K^{-1}(p_K^*)$ is tedious, and it is reported in Appendix C. The resulting expression for t^{D^*} is:

$$t^{D^*} = \begin{cases} \dfrac{q_0 L + 3\sigma_H p_K + [(q_0 L + 3\sigma_H p_K)^2 - (q_0 + p_1)(q_0 - p_1)L^2]^{1/2}}{Q(q_0 - p_{1\prime})} \\ \qquad 0 < p_K < \dfrac{\mu_H p_1 + 1.5\sigma_H q_0}{\mu_H + 1.5\sigma_H} \\ \dfrac{\mu_H(q_0 - p_1)}{Q(q_0 - p_K)} \quad , \quad \dfrac{\mu_H p_1 + 1.5\sigma_H q_0}{\mu_H + 1.5\sigma_H} \leq p_K < q_0 \\ \infty \quad , \quad q_0 \leq p_K < 1 \end{cases},$$

where

$$\begin{aligned} q_o &= 1 - p_o \\ L &= \mu_H - 1.5\sigma_H \ . \end{aligned} \tag{3.31}$$

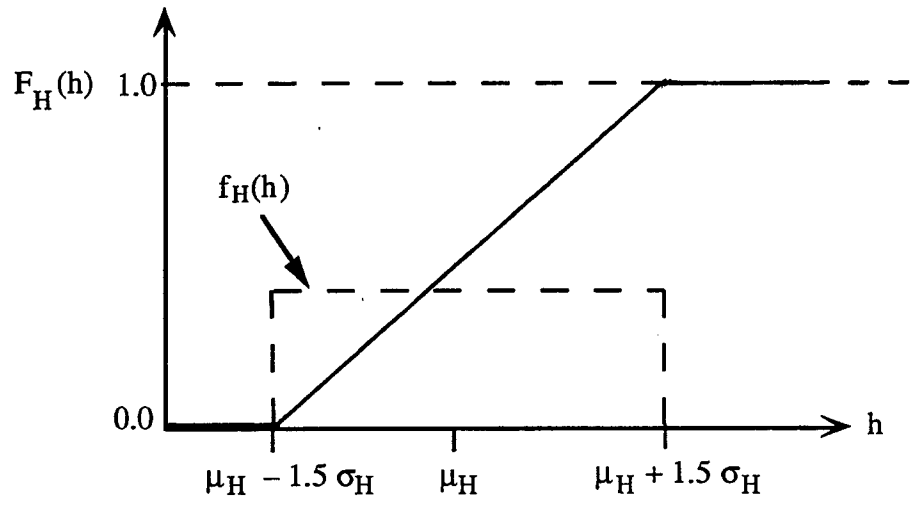

Figure 3.12. For our problem, this is a good approximation to the Gaussian cumulative distribution.

3.4 Designing Smooth Command Schedules to Minimize Hardware Jerk

To the platform pointing controller, a target schedule is simply a sequence of step inputs. Unless some smoothing or shaping is applied to these commands, various vibrational modes could be excited on the platform, resulting in unacceptable increases in settling time of the beam control hardware, the platform forebody in particular. But a brief glance at the typical threat distribution of Fig. 3.13 suggests that an effective fast steering and forebody control law could be developed where the lighter fast steering subsystem could be used to direct the beam from target to target, while the heavier forebody and the main body slowly drift through the target, thereby minimizing platform jerking.

Such a control law must simultaneously solve two optimization problems—a geometry problem and a timing problem. The geometry problem is to find an optimal path through the target set, without consideration of time. The *time* problem is to find an optimal *trajectory* through the target set by associating optimal times with the various points along the path. This process is obviously subject to the constraint that fast steering and forebody motions are limited, as shown by the circles in Fig. 3.13, the radius of each circle indicating the maximum fast steering subsystem deflection allowed from the center, where the forebody is aimed (main body ignored here). The problem then essentially reduces to sweeping through the targets set with a moving circle in such a way that the center of the circle is moving smoothly.

Let X^T_{k+r} be the position of the r_{th} target following T_k in the target sequence, and let t_{k+r}be the time at which processing of T_{k+r} starts. Then t_{k+r} is the sum of t_k and all the dwell and retarget times for the targets between T_k and T_{k+l}:

$$t_{k+r} = t_k + \sum_{m=0}^{r-1} t^D_{k+m} \,. \tag{3.32}$$

Using the target sequence and the processing times t_{k+r}, our recursive approach does *not* generate position (angular displacement step function) commands. Instead, it produces smooth velocity commands, expressed as displacement-time pairs $(\Delta X, \Delta t)$ as follows.

START

1. Assume velocity command output at time t_k is $(X^T_{k+r} - X^{FB}_k,\ t_{k+r} - t_k)$.

2. Change velocity command to $(X^T_{k+r+1} - X^{FB}_k,\ t_{k+r+1} - t_k)$ if all targets between T_k and T_{k+r+1} can be reached by redirecting *only* the fast steering subsystem when the forebody is moving along the straight-line segment from X^{FB}_k to X^T_{k+r} at a

constant velocity

$$\dot{X}^{FB} = \frac{X^T_{k+r+1} - X^{FB}_k}{t_{k+r+1} - t_k}$$

3. Otherwise, do not change the velocity command, and retry $(X^T_{k+r+2},\ t_{k+r+2})$ when processing of T_k is complete.

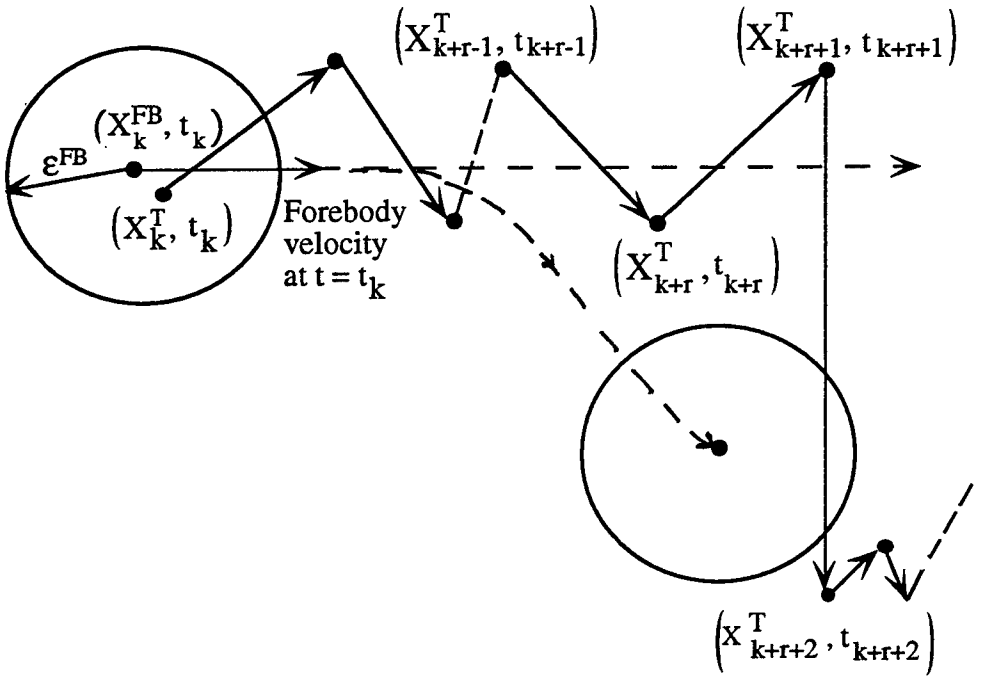

Figure 3.13. The forebody trajectory from (X^{FB}_k, t_k) towards $\left(X^T_{k+r+1},\ t_{k+r+1}\right)$ may be extended towards $(X_{k+r+2},\ t_{k+r+2})$ (as shown) if all intervening targets fall within the tube generated by the moving circles (each node is a target state-time pair).

As seen from the dashed lines in Fig.3.13, the extension works for $(X^T_{k+r+1},\ t_{k+r+1})$ but fails for the next pair $(X^T_{k+r+2},\ t_{k+r+2})$ due to the excessive forebody angular displacement required. Except for such extreme—and infrequent—displacements, the algorithm continuously updates its velocity output as targets are processed and new targets come within the projected reach of the forebody. For the unfavorable example of Fig. 3.13, the output signal produces velocity outputs whose direction is indicated by the unit vectors assigned to each target vertex. The forebody trajectory is thus a smooth path through target ϵ^{FB}-neighborhoods, as shown in Fig. 3.14, and Fig. 3.15 exhibits the flow chart for the smoothing algorithm. Observe the following definitions for that chart:

$$X^{FB}(t_m) = X^{FB}(t_k) + \frac{X^T_{k+r}(t_{k+r}) - X^{FB}}{(t_{k+r} - t_k)}(t_m - t_k)\,, \tag{3.33}$$

where t_{k+r} is the start time of T_{k+r} derived from Eq. (3.32).

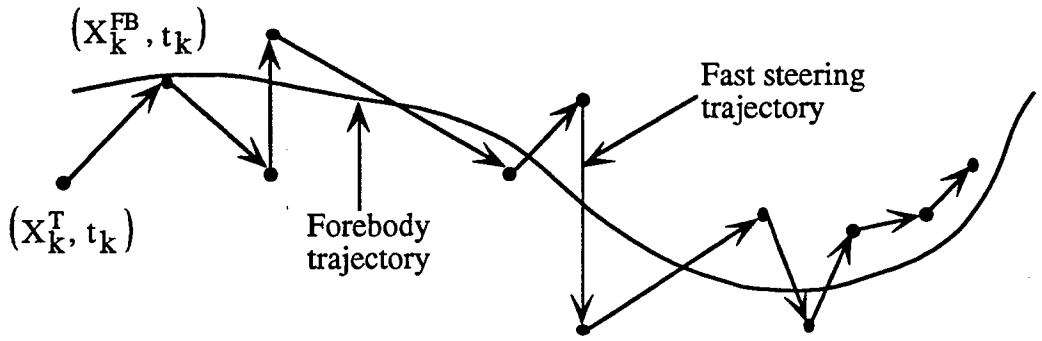

Figure 3.14. The forebody drifts smoothly through target ϵ^{FB}-neighborhoods.

To quantitatively estimate the smoothing characteristics of this subroutine, recall some elementary notions from second-order linear systems theory (Kuo 1975). For linear systems, "jerk" is defined as the third derivative $c^{(3)}(t)$ of the system output c(t). For a second-order critically damped system with step input $U(t)$ of size U_o ,

$$c(t)_u = U_0(1 + \omega_n t e^{-\omega_n t})\,,$$
$$c^{(3)}(t)_u = U_0 \omega_n^3 e^{-\omega_n t}[3 - \omega_n t]$$

whose maximum occurs at $t = 4/\omega_n$ and equals

$$\left(c^{(3)}(t)_u\right)_{\max} = -e^{-4}\omega_n^3 U_0\,. \tag{3.34}$$

For a ramp input R of size R_0,

$$c(t)_R = R_0\left[t - \frac{2}{\omega_n} + te^{-\omega_n t}\right]\,,$$
$$c^{(3)}(t)_R = R_0\omega_n^2\left(3 - \omega_n t\right)e^{-\omega_n t}\,,$$

whose maximum also occurs at $t = \frac{4}{\omega_n}$, and equals

$$\left(c^{(3)}(t)_u\right)_{\max} = -e^{-4}\omega_n^2 R_0\,. \tag{3.35}$$

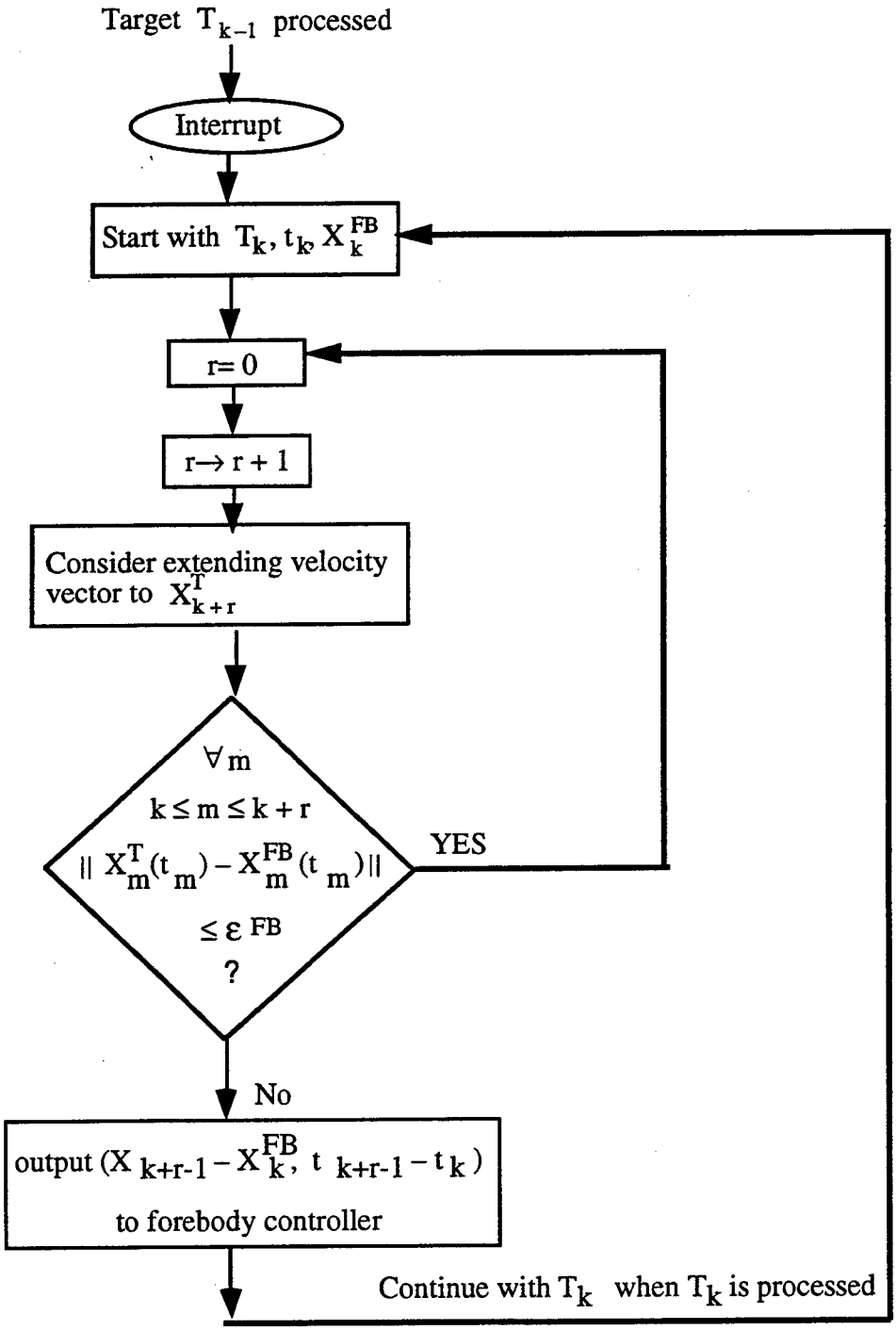

Figure 3.15. Flowchart for the forebody smoothing algorithm.

The ratio of step jerk to ramp jerk is

$$A_{\text{jerk}} = \frac{\omega_n^{FB} U_0}{R_0} ,$$

where ω_n^{FB} is the characteristic frequency of the forebody.

Now consider a target sequence $\mathcal{T} =< T_1, \ldots, T_i, \ldots T_N >$ whose target-to-target angular displacements are $U =< \theta_1, \ldots, \theta_i, \ldots \theta_{N-1} >$. Then, without smoothing, if we define $\theta^* = \max \{\theta_i : i = 1, \ldots, N-1\}$, the forebody jerk magnitude may reach $(\omega_n^{FB}))^3 \theta^* e^{-4}$. But, using the smoothing algorithm of Fig.3.15, the *smoothed* jerk magnitude is at most $(\omega_n^{FB})^3 R_{\max} e^{-4}$, where

$$R_{\max} = \max \left[\begin{array}{ll} \dfrac{\theta_{k_1,k_2}}{\sum_{r=k_1}^{r=k_2} (t_r^R + t_r^D)} & : \quad \{T_{k_1}, \ldots, T_{k_2}\} \subset \mathcal{T} \text{ and} \\ & \| X_k^T(t_k) - X^{FB}(t_k) \| \leq \epsilon^{FB} , \\ & k = k_1, \ldots, k_2 . \end{array} \right]$$

$$= \frac{\theta_{k_1^*,k_2^*}}{\sum_{r=k_1^*}^{k_2^*} (t_r^R + t_r^D)} .$$

The jerk attenuation ratio is thus at least

$$A_j = \frac{\omega_n \, \theta^*}{R_{\max}} . \tag{3.36}$$

For many reasonable threats—especially for target-rich ones—the angular distance θ_{k_1,k_2} will be small compared to the sum $\sum_{r=1}^{k_2-k_1} \theta_{k_r,k_r+1}$ of intervening distances, and $k_2 - k_1$ will be close to the total quantity N of targets. In such cases, the jerk attenuation ratio will be very large, possibly infinite, since the forebody may not have to be moved at all. Consider a more conservative example where

$$\begin{aligned} \text{N} &= 100 \\ k_1^* - k_2^* &= \frac{N}{2} \\ \omega_n^{FB} &= 10 \text{ sec}^{-1} \\ t_r^R &= 0.1 \text{ sec} \\ t_r^D &= 0.3 \text{ sec} \\ \theta^* &= 10 \text{ mrad} \end{aligned}$$

$\theta_{k_1^*,k_2^*} = 50$ mrad

Then

$$R_{\max} = \frac{50 \text{ mrad}}{50 \times 0.4 \; sec} = 2.5 \text{ mrad/sec} \; ,$$

and

$$A_j = \frac{10 \times 10 \text{ mrad/sec}}{2.5 \text{ mrad/sec}} = 40,$$

a considerable reduction in forebody jerk.

4. The Target Scheduling Algorithm for the Boost Phase

4.1 Introduction

In previous chapters, we formulated the Target Scheduling Problem (TSP) as a statistical decision problem consisting of three major subproblems, and in Section 2.2.2.2 and Chapter 3, we provided a functional overview of DDTS.

Consider the threat illustrated in Fig. 4.1, where T_{ij} is the j^{th} target to be processed by deadline d_i. Although we address the boost phase in this section, we should note that during midcourse the point-targets shown as dots in Fig. 4.1 represent "clouds" of objects. These clouds contain various mixtures of Re-entry Vehicles (RVs), decoys, and replicas, and we shall examine these representative dots in greater detail in Chapter 5. Observe that the threat pictured in Fig. 4.1 is rather benign, with clearly separated deadline classes. In general, most complex threats must be assumed to contain targets whose positions are not as conveniently correlated with their deadline.

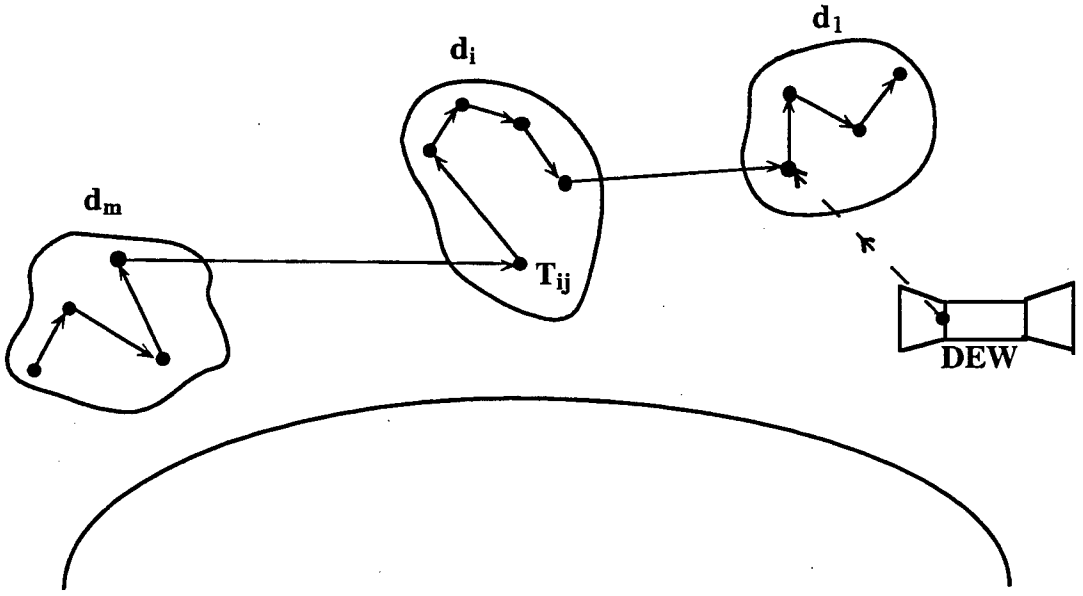

Figure 4.1. A threat with m deadlines showing the m deadline-equivalence classes.

In this chapter, we focus on algorithmic and computational issues. As before, we emphasize ground-based and space-based directed energy weapons in the boost phase, deferring to Chapter 5 any extensions to the midcourse discrimination phase. We only discuss the principal flows of the DDTS algorithm, avoiding unessential details about its software implementation. While we provided a brief version of "heaps" to illustrate

how paths and tours are manipulated in the computer, we have deliberately omitted any discussion of pointer arithmetic, dynamic memory allocation, and other powerful features of the C-language used in our program. These and other implementation details can easily be extracted from C-language users' manuals and from the source code of the DDTS program.

Viewed as a computational module, DDTS is a multiple-input-multiple output system specified by a large number of parameters. Referring to the simplified description in Fig. 4.2, the threat information consists of a *threat assignment* $\mathcal{T} = \{T_i : i = 1, \ldots, N\}$, each target (label) T_i a vector specifying the position and velocity vector, deadline and release time, type, vulnerability radius, aspect angle, hardness mean and variance, and value of the target.

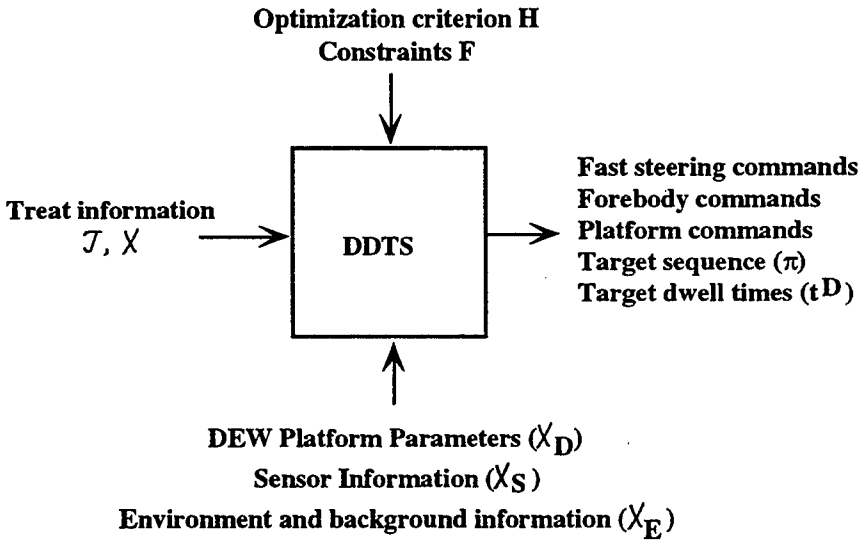

Figure 4.2. Input-output sketch of DDTS.

The DEW platform is specified by an Acquisition, Tracking, and Pointing bias and variance, a pointing state, a "field of action" (field of regard), position and slew saturation levels, damping constants, and desired settling accuracies for the fast steering subsystem, the forebody, and the main platform body. The DEW itself is defined by its beamwidth, its frequency, its power, and its bias and jitter. The environment is specified by the engagement geometry and the relative state of the DEW and each target, but no specific background information is considered in the current version of the algorithm.

Sensors are characterized by their position and velocity vectors, their beamwidths, and their tracking errors. The target ID information provided by the sensors is represented as a discrimination matrix $[p_{ij}]$ whose entries p_{ij} specify the probability that a target of type j will be classified as one of type i.

The optimization criterion H defines the measure by which the performance of DDTS will be judged. For target scheduling in the boost phase, this measure is essentially the same as that used in judging the overall platform performance, since target scheduling is the principal platform management task. Consequently, we use expected target leakage risk as a criterion during the boost phase. As we discussed in earlier sections, minimizing this risk is a complex optimization process subject to various constraints like deadlines, release times, and energy limits, and these are specified in the constraint structure F.

To provide a more detailed explanation of how DDTS computes an optimal schedule, we discuss the algorithm from a perspective where the problem is broken into just two major parts, a time optimization and a p_K optimization, as shown in the two-level approach of Fig. 3.1.

The lower-level inner loop is a "local" optimization subroutine where, for each p_K considered in the outer loop, target schedules are found that reduce target completion times until no tardy targets remain or until minimum completion times are attained. In the event that some tardiness remains, least important targets are rejected in accordance with the rejection rule r (Eq. (3.7)) until all targets scheduled for processing can meet their deadline.

This time-line optimization is usually not sufficient, of course, since total leakage risk typically *increases* as dwell times are reduced past a certain minimal value. To account for this time-lethality trade-off, DDTS includes an outer loop where total leakage risk is minimized by finding the optimal value of p_K using the BRENT Algorithm (Brent 1973; Press et al. 1988).

4.2 Local Optimization: Optimal Sequencing and Target Rejection

Four major functions are performed in the inner optimization loop (see Fig. 4.3):

1. Initial tour construction.
2. Tour verification.
3. Tour improvement.
4. Target rejection.

Even though dwell times are not optimized in the inner loop, their values are essential in the computation of target completion times and tour time-lines. In each of the tour calculations, dwell times were obtained from p_K values fixed in the outer loop.

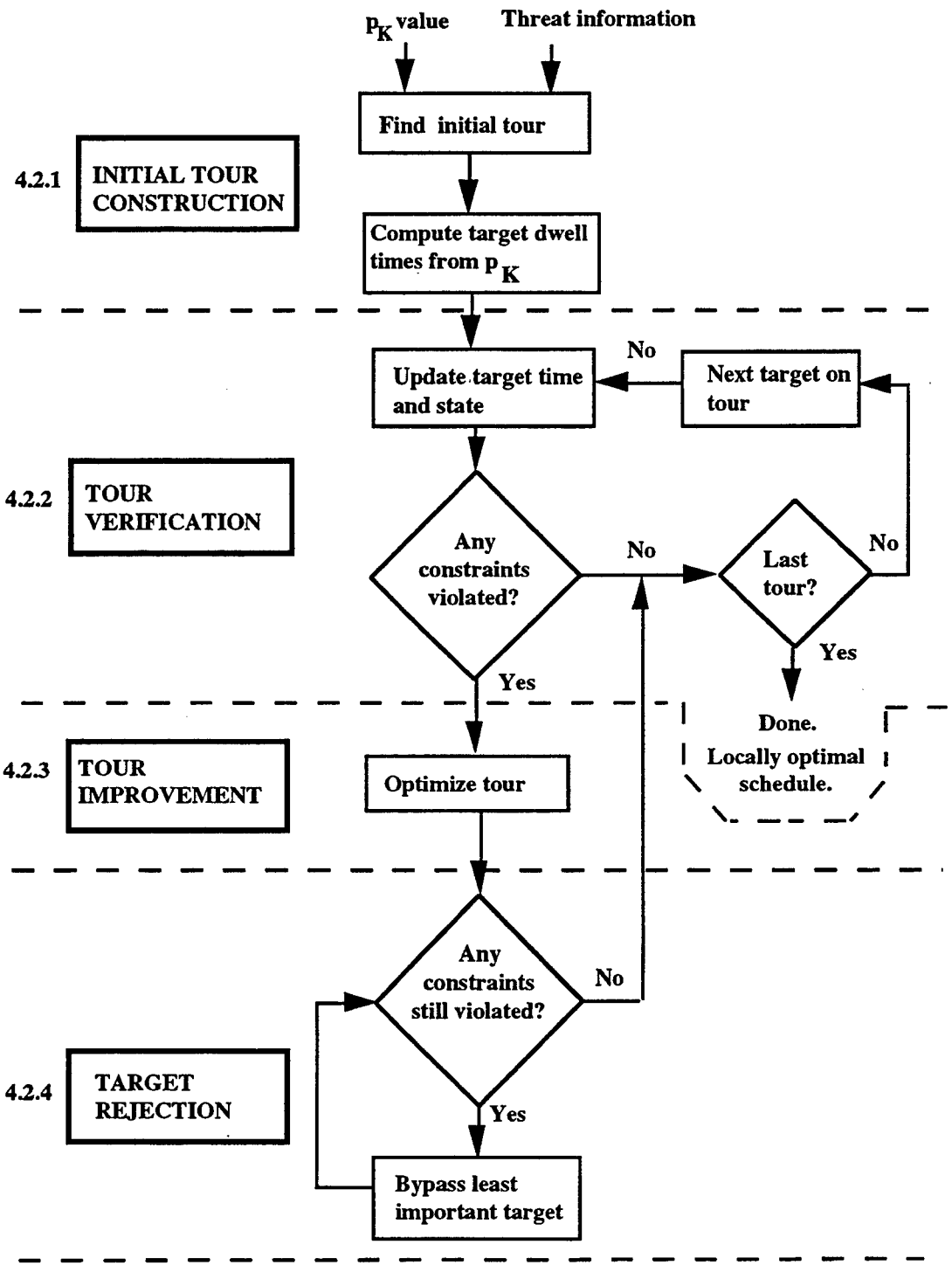

Figure 4.3. The inner loop of the DDTS algorithm is a local optimization loop with four major subroutines.

4.2.1. Initial Tour Construction (Subroutine INITOUR)

Using appropriate kinematic information about the targets and a fixed p_K value received from the outer loop, the first operation in the inner loop is to construct an initial tour through the threat. Considering that all the targets are moving, this is an interactive process which alternates between the three major tour construction steps described in Chapter 3 and a target update process where state updates are computed as a tour is developed. A more detailed view of INITOUR, the algorithm that actually finds the initial tour, is provided in Fig. 4.4.

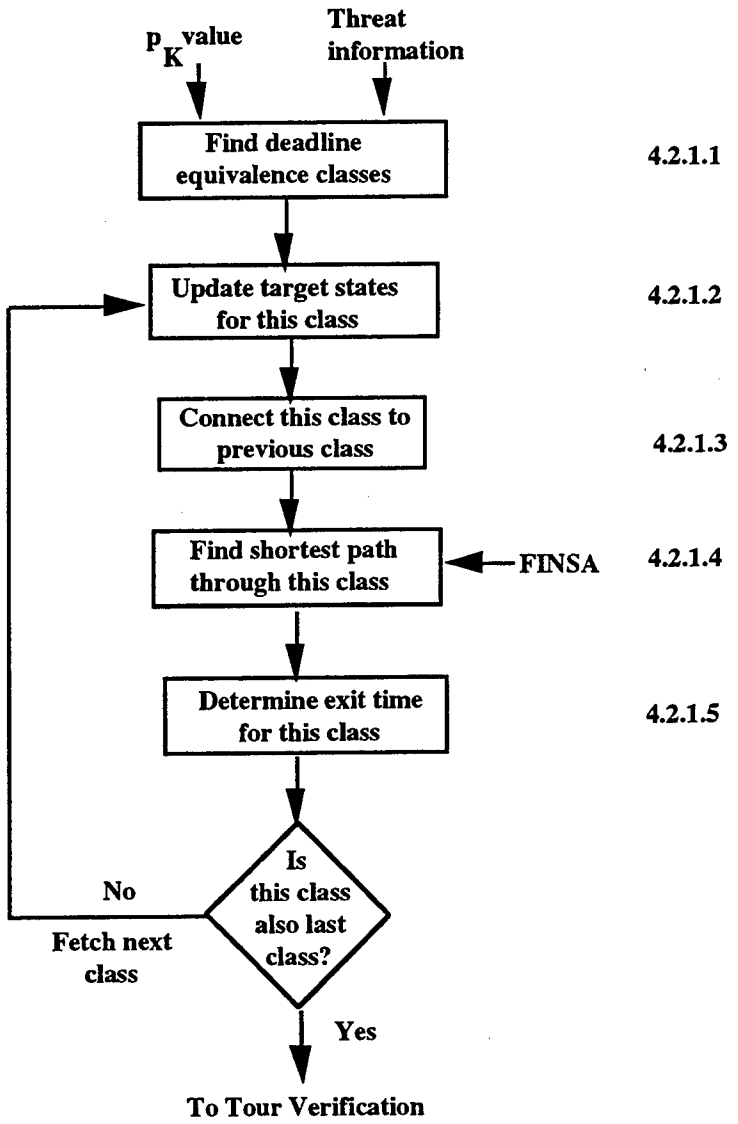

Figure 4.4. The INITOUR initial tour construction subroutine.

4.2.1.1. Computing Deadline Equivalence Classes. The first step in conducting a tour is to construct and link *deadline equivalence classes,* sets of targets that share the same deadline. To further refine the deadline partition and the ordering of the targets, deadline-equivalent targets are further ranked in accordance with their release or availability times. Hence, two targets are equivalent if they have the same deadline and release time (lexicographic order (Aho et al. 1974)).

In the DDTS algorithm, target classes are linked lists of objects, and class construction involves several standard operations: sorting, ordering, inserting, deleting and searching. Because there may be many deadline equivalence classes, and each may have to be recomputed many times due to the insertion or deletion of targets throughout an engagement, the computational efficiency of the class construction process is essential. We have chosen a *heap** as the basic data structure for doing this job. Let us briefly review heaps and their use.

4.2.1.1.1 Heaps. Consider a scalar ordering $\leq$ on the real numbers and a set $H = \{a_i : i = 1, \ldots, N\}$ of N numbers. Then H is a *heap* (see Fig. 4.5), if its elements satisfy the relation

$$a_{\lfloor j/2 \rfloor} \leq a_j, 1 \leq \lfloor j/2 \rfloor \leq N, \tag{4.1}$$

where $\lfloor x \rfloor$ is the smallest integer greater than x. A more detailed but informal discussion of heaps was taken from Press et al. (1988) and is provided in Appendix D.

Sorting with heaps is done in two major steps.

1. Heap Construction.
2. Heap Search.

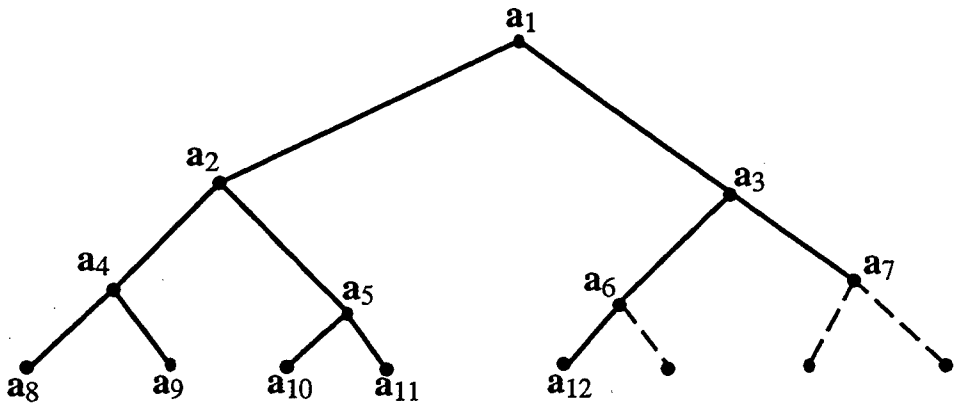

Figure 4.5. A 12-element heap and its implied ordering. Elements are ordered only "vertically" and not "laterally".

* Press et al. (1988); Aho et al. (1974); Knuth (1973); Sedgewick (1988).

4.2.1.1.2 Heap Construction. Depending upon whether the set H of data is available as a whole (concurrently) or only one element a_i at a time (sequentially), an algorithm like HEAPIFY (Horowitz and Sahni 1978) (also BUILDHEAP (Aho et al. 1974)) or INSERT (Horowitz and Sahni 1978) is used, respectively. The latter constructs a heap one element at a time, requiring O(n log n) work, whereas the former benefits from having all the data at once and only requires $O(n)$ to construct the heap. Due to constant target updates (new targets or destroyed targets), DDTS uses *insertion* to construct a heap.

4.2.1.1.3 Heap Search. A heap only structures the data and does not by itself provide a total ordering. To get the ordering (sort), elements must be removed from the heap one at a time, largest element first. This requires $O(logn)$ work per element, or $O(nlogn)$ for the entire heap.

4.2.1.1.4 HEAPSORT. Both construction and search steps have been combined into an algorithm known as HEAPSORT.* When the set $H = \{a_i : i = 1, \ldots, N\}$ of input objects is available simultaneously, heap construction is easiest, and a worst case of only $O(NlogN)$ work is required by HEAPSORT to sort the objects. In the sequential case where element-by-element insertion is required, $O(2NlogN)$ may be needed (Horowitz and Sahni 1978). Other algorithms have been developed to sort data quickly, but only QUICKSORT is worth mentioning here. On the average, QUICKSORT actually requires only $O(NlogN)$, a factor of 2 better than the worst case for HEAPSORT, which also requires about as much on the average. But the fact that the average and the worst case behavior of HEAPSORT were comparable and that QUICKSORT is actually an $O(N^2)$ algorithm in the worst case were sufficient reasons to choose HEAPSORT for our problem because there is no evidence that many threat scenarios will be "average" in any sense.

4.2.1.2. Updating Target States. We define the *exit time* of a class as the completion time of the class or, equivalently, the completion time of the last element of the class. The exit time t_e of an n-target class is thus

$$t_e = t_e^0 + \sum_{i=1}^{n} t^R_{\pi(i)} + \sum_{i=1}^{n} t^D_{\pi(i)} \,, \tag{4.2}$$

where, in accordance with the retarget and dwell time calculations of Sections 2.2.1.3 and 2.3.3.1,

$t^R_{\pi(i)}$ is the retarget time to Target $\pi(i)$ in position i,
$t^D_{\pi(i)}$ is the dwell time of Target $\pi(i)$ in position i, and
t_e^o is the exit time of the previous class.

* Knuth (1973); Sedgewick (1988); Horowitz and Sahni (1978).

Even though the state of each target is time-varying, only one clock time is assigned to each equivalence class and to its target members during initial tour construction. This provides considerable computational economy and will be checked later during tour verification. Using the same definitions as in Eq. (4.2), the *clock time* $\bar{t}_c$ of a class is the "average" clock time of its members, also called the *class time*:

$$\bar{t}_c = t_e^o + \frac{1}{2}\Big(\sum_{i=1}^{n} t^R_{\pi(i)} + \sum_{i=1}^{n} t^D_{\pi(i)}\Big) \,. \tag{4.3}$$

Using this clock time and the initial states of all targets in the class, Eqs. (2.29–2.32) can be used to calculate the states of targets at exit times and at all future class times.

4.2.1.3. Connecting Target Classes. Target class connection is done by recursion. A path through a target class starts at a target determined by the previous class and ends at the target chosen to minimize the total retarget time from the starting target to the first target in the next class. Consider the three contiguous classes shown in Fig. 4.6. If T_1 is the starting target in C_2 (determined by the topology of C_1) then, given the tour shown in C_2, T_a and T_b are the only two potential successors to T_1, and only two paths are allowed through C_2, $\pi_1 =< T_1,\ T_b,\ T_x,\ T_a >$ and $\pi_2 =< T_1,\ T_a,\ T_x,\ T_b >$. Whether π_1 or π_2 is chosen is determined by the following cost comparison: connect T_a to T_c if

$$t^R(T_a, T_c) - t^R(T_1, T_a) \leq t^R(T_b, T_d) - t^R(T_1, T_b), \tag{4.4}$$

otherwise connect T_b to T_d.

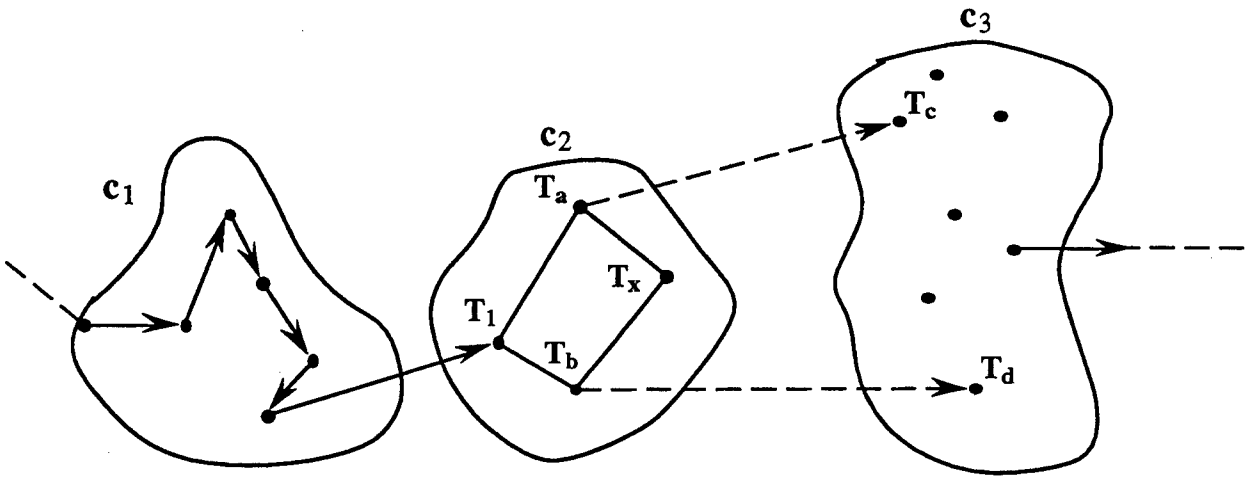

Figure 4.6. Classes are connected to minimize retarget time.

4.2.1.4. Shortest Path Through a Class. In the previous section, a shortest path through class C_2 was assumed. As discussed in detail in Section 3.1, this path is found using the Farthest Insertion Algorithm (FINSA).

4.2.1.5. Class Exit Target and Exit Time. The exit *target* is found as shown in Section 4.2.1.3. The exit *time* was derived in Eq. (4.2), where it was also called the *completion time.*

4.2.2. Tour Verification

An initial path through the target set was developed using the INITOUR subroutine, assuming that all the targets in any deadline equivalence class have the same clock time. During *tour verification*, the validity of this assumption is examined by verifying that all the deadline and energy constraints are met along the chosen path, as shown in Fig. 4.3. If they are, nothing remains to be done. If not, an improved path through the targets must be found, and this is the subject of the next section.

4.2.3. Tour Improvement

In Section 3.1.3, we motivated the need for tour improvements, and we presented a brief discussion of the 2-opt procedure. Now we provide a simple flowchart of how 2-opt is used in the DDTS algorithm (Fig. 4.7).

The tour improvement subroutine is simple. For each candidate edge-pair $(e_k,\ e_q)$ there must be at least a *local timeline* improvement (local to the pair $(e_k,\ e_q)$ before testing whether the constraints are still met for all targets T_m preceding T_k, the last target whose position was altered by 2-opt. Note that the routine is a bit more complicated than shown, due to the recursive nature of path improvements. As paths are improved, new collections of pairwise improvements appear, and q_{max} and k_{max} must be reset periodically. To avoid combinatorial "runaway", only a given number of iterations can be afforded, and this "maximum count" is set adaptively as a point of marginal diminishing returns is approached.

One effective way to control runaway is to limit the range of the 2-opt process by limiting q_{max} and k_{max} to only a fraction of the quantity of targets preceding the first late target T_j.

Due to release times, a marginal timeline improvement may not assure a global timeline improvement. Only when a global improvement is assured is the pair $(e_k,\ e_q)$ accepted and is it "cemented" in the tour. But the first late target T_j that initiated 2-opt may still be late, and the process may continue until the maximum count is reached.

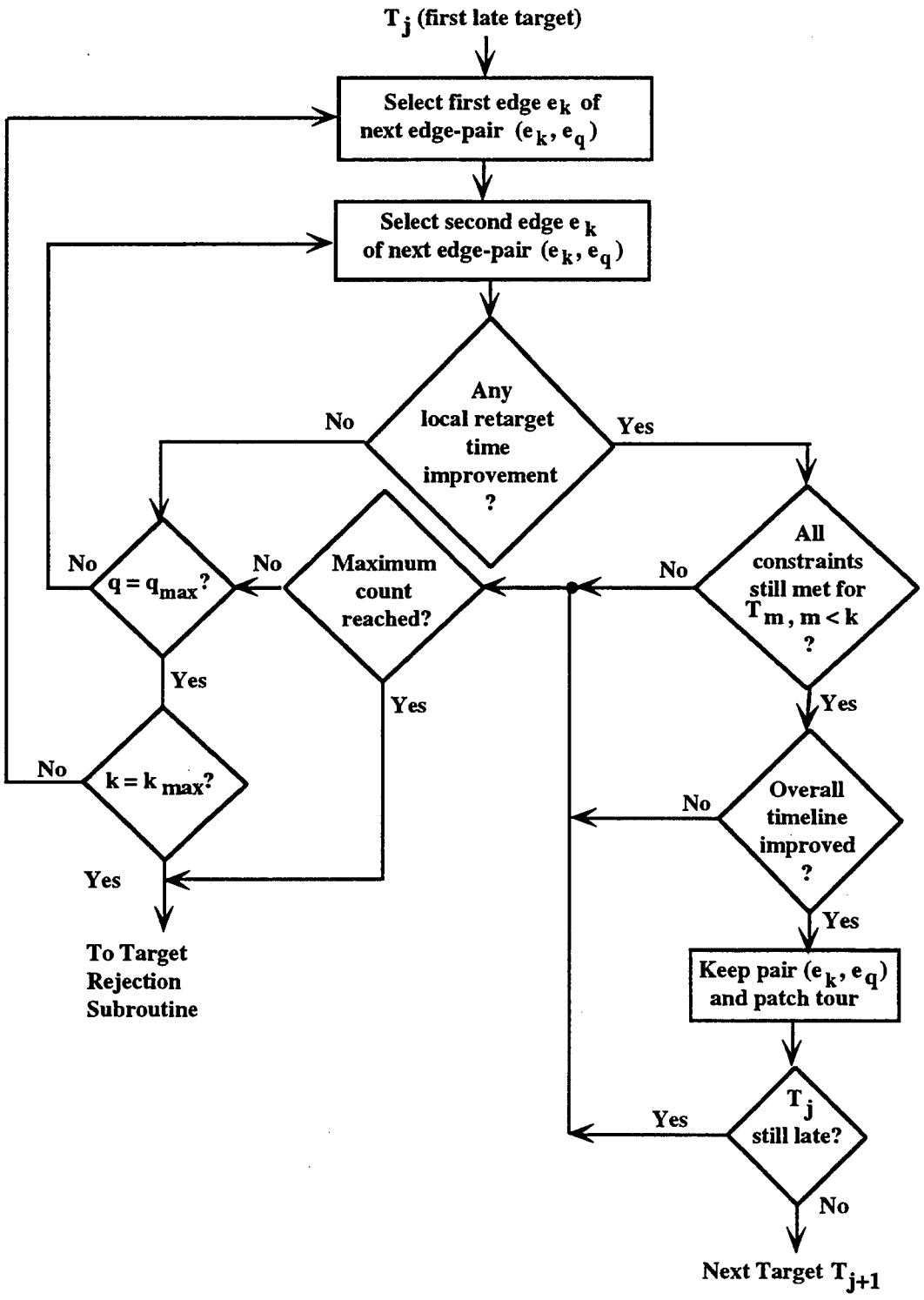

Figure 4.7. Simplified flowchart of the 2-opt tour improvement subroutine.

4.2.4. Target Rejection

If T_j is still late after the maximum allowable number of pairwise exchanges, one or more targets must be ignored, as shown in Figs. 4.3 and 4.8. This is accomplished by organizing all targets into a heap—the "r-heap"—using the rejection ratio r of Eq. (2.50) as a criterion, the target with lowest r-value considered as the maximal heap element (see Section 4.2.1.1.1 for a more detailed discussion of heaps). O(log N) work is required to pull the maximal target from the r-heap, and another O(log N) work to "reheap" the remaining targets. Returning to Fig. 4.8, this target rejection process must halt since there is at least one target that satisfies its deadline constraint.

As a closing comment for this subsection, note that the rotation and position extensions described above must be executed repeatedly as new target sequences and schedules are tried during the optimization process. To avoid unnecessary repetitions of these calculations, the future positions of all the targets are calculated just once and stored in a look-up table for further reference. Further computational gains are made by defining these future target states on a time scale of appropriate granularity, typically one second.

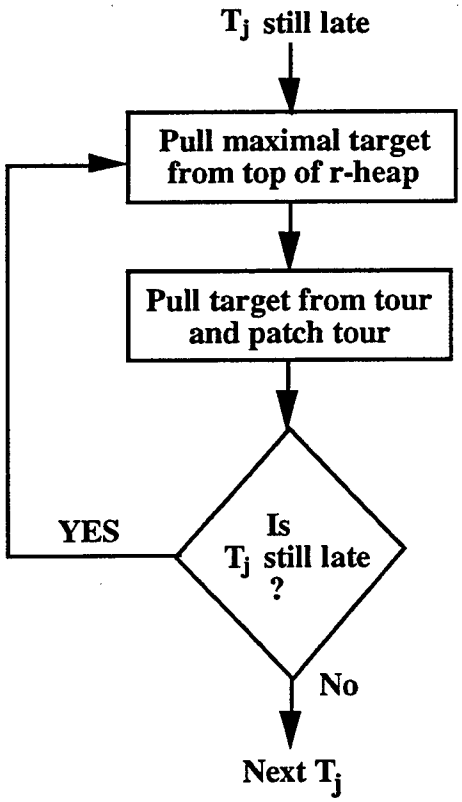

Figure 4.8. Target rejection is accomplished using the r-heap.

4.3 Global Optimization: Optimal Dwell Times

The outer loop of DDTS shown in Figs. 2.5 and 3.1 minimizes the leakage risk by finding an optimal value of p_K. By referring to this process as "global optimization" we are abusing conventional terminology somewhat,* and we should actually refer to it as *outer optimization* since we have in effect a function $f(x, y, z)$ of three variables for which we seek a minimum,

$$f^* = \min_{x,y,z}\Big\{f(x,y,z)\Big\} = \min_{x}\ \Big\{\min_{y}\ \Big\{\min_{z}\ \{f(x,y,z)\}\Big\}\Big\}\ , \tag{4.5}$$

where x is the *outer variable representing* p_K. This outer loop is a scalar optimization loop whose convexity properties were discussed in Section 2.2.2, where we showed that, as a function of p_K, the objective function $\mathcal{R}(p_K)$ is a jagged function that exhibits several local minima, potentially one for each rejected target. But, unless deadlines are extremely hard and the threat environment is target-poor, jaggedness is not severe and the risk function behaves in a relatively smooth and convex manner. So it is reasonable to consider a golden section search (Press et al. 1988) since it is designed to handle the worst possible case of function minimization. But if the function is nicely parabolic near the minimum, then the parabola fitted through any three points should drive us in a single leap to a point very near the minimum. Since an abscissa is sought rather than an ordinate, the procedure is technically called *inverse parabolic interpolation* (Fig. 4.9).

As Press et al. (1988) explain it, the formula for the abscissa x which is the minimum of a parabola through three points $f(a)$, $f(b)$, and $f(c)$ is

$$x = b + \frac{1}{2}\frac{(b-a)^2[f(b)-f(c)] - (b-c)^2[f(b)-f(a)]}{(b-a)[f(b)-f(c)] - (b-c)[f(b)-f(a)]}. \tag{4.6}$$

This formula fails only if the three points are collinear, in which case the denominator is zero (minimum of the parabola is infinitely far away). Note, however, that Eq. (4.6) is as happy jumping to a parabolic maximum as to a minimum. No minimization scheme that depends solely on Eq. (4.9) is likely to succeed in practice.

The exacting task is to use a scheme which relies on a sure-but-slow technique, like golden section search, when the function is not cooperative, but which switches over to Eq. (4.6) when the function allows. The task is nontrivial for several reasons, including these: (1) The housekeeping needed to avoid unnecessary function evaluations in switching between the two methods can be complicated. (2) Careful attention must be given to the "endgame," where the function is being evaluated very near to the roundoff limit of Eq. (4.6). (3) The scheme for detecting a cooperative versus noncooperative function must be very robust.

* Papadimitriou and Steiglitz (1982); Press et al. (1988).

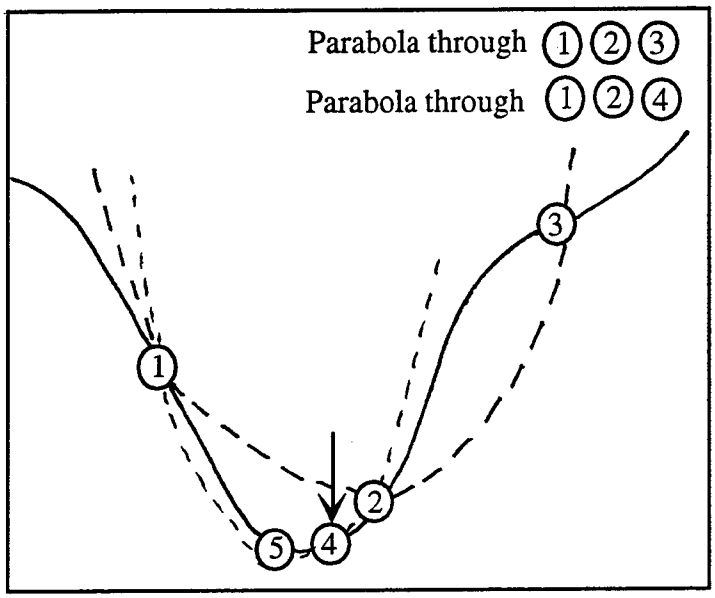

Figure 4.9. Convergence to a minimum by inverse parabolic interpolation. A parabola (dashed line) is drawn through the three original points 1,2,3 on the given function (solid line). The function is evaluated at the parabola's minimum, 4, which replaces point 3. A new parabola (dotted line) is drawn through points 1, 4, 2. The minimum of this parabola is at 5, which is close to the minimum of the function.

Brent's method (Brent 1973) is up to the task in all particulars. At any particular stage, it is keeping track of six function points (not necessarily all distinct), a, b, u, v, w, and x, defined as follows: the minimum is bracketed between a and b; x is the point with the very least function value found so far (or the most recent one in case of a tie); w is the point with the second least function value; v is the previous value of w; u is the point at which the function was evaluated most recently. Also appearing in the algorithm is the point x_m, the midpoint between a and b; however, the function is not evaluated there.

You can read the code (Brent 1973) to understand the method's logical organization. Mention of a few general principles here may, however, be helpful: Parabolic interpolation is attempted, fitting through the points x, v, and w. To be acceptable, the parabolic steps must (1) fall within the boundary interval (a, b) and (2) imply a movement from the best current value x that is less than half the movement of the *step before last.* This second criterion insures that the parabolic steps are actually converging to something, rather than, say, bouncing around in some noncovergent limit cycle. In the worst possible case, where the parabolic steps are acceptable but useless, the method will approximately alternate between parabolic steps and golden sections, converging in due course by virtue of the latter. The reason for comparing to the step

before last seems essentially heuristic: experience shows that it is better not to "punish" the algorithm for a single bad step if it can make it up on the next one.

Another principle exemplified in the code is never to evaluate the function less than a distance TOL from a point already evaluated (or from a known bracketing point). The reason is that, as we saw in Eq. (4.6), there is simply no information content in doing so: the function will differ from the value already evaluated only by an amount of order ϵ, the runoff error or function jaggedness. Therefore in the code there are several tests and modifications of a potential new point, imposing this restriction. This restriction also interacts subtly with the test for "doneness," which the method takes into account.

A typical ending configuration for Brent's method is that a and b are $2 \times x \times TOL$ apart with x (the best abscissa) at the midpoint of a and b, and therefore fractionally accurate to $\pm TOL$.

Note that TOL should generally be no smaller than the square root of the computer's floating point precision, and that TOL can be set sufficiently high that, within acceptable risk errors, local minima are ignored as feasible solutions, and only global minima are obtained.

As concluding comment on the outer optimization loop, observe that dwell times t^D are controlled and specified by values of p_K. The relationship between the t^D vector and p_K is monotonic and is formally derived in Appendix B.

4.4 The Ground-Based DEW

We have focused thus far on DEW systems that are *space-based.* When laser energy is derived from a ground-based station, a relay or collection subsystem must be added to the DEW system, and additional articulation constraints must be introduced. Referring to the hypothetical design of Fig. 4.10, the DEW platform must remain oriented so that proper alignment can be maintained between the collector ("catcher") C, the fighting mirror FS, the target T, the base B, and the forebody FB. If we define the vectors as in Fig. 4.10, we have the following angular constraint on the forebody-to-catcher angular position θ^{FC}:

$$\theta^{FC} \geq \pi - \theta^{FB}_{max} - \theta^{C}_{max} \,, \tag{4.7}$$

or

$$\frac{X^E_{DT} \cdot X^E_{DB}}{\| X^E_{DT} \| \| X^E_{DB} \|} \leq cos\left(\pi - \theta^{FB}_{max} - \theta^{C}_{max}\right) . \tag{4.8}$$

The constraint is thus

$$\frac{X^E_{DT} \cdot X^E_{DB}}{\| X^E_{DT} \| \| X^E_{DB} \|} \leq -cos\left(\theta^{FB}_{max} + \theta^{C}_{max}\right) . \tag{4.9}$$

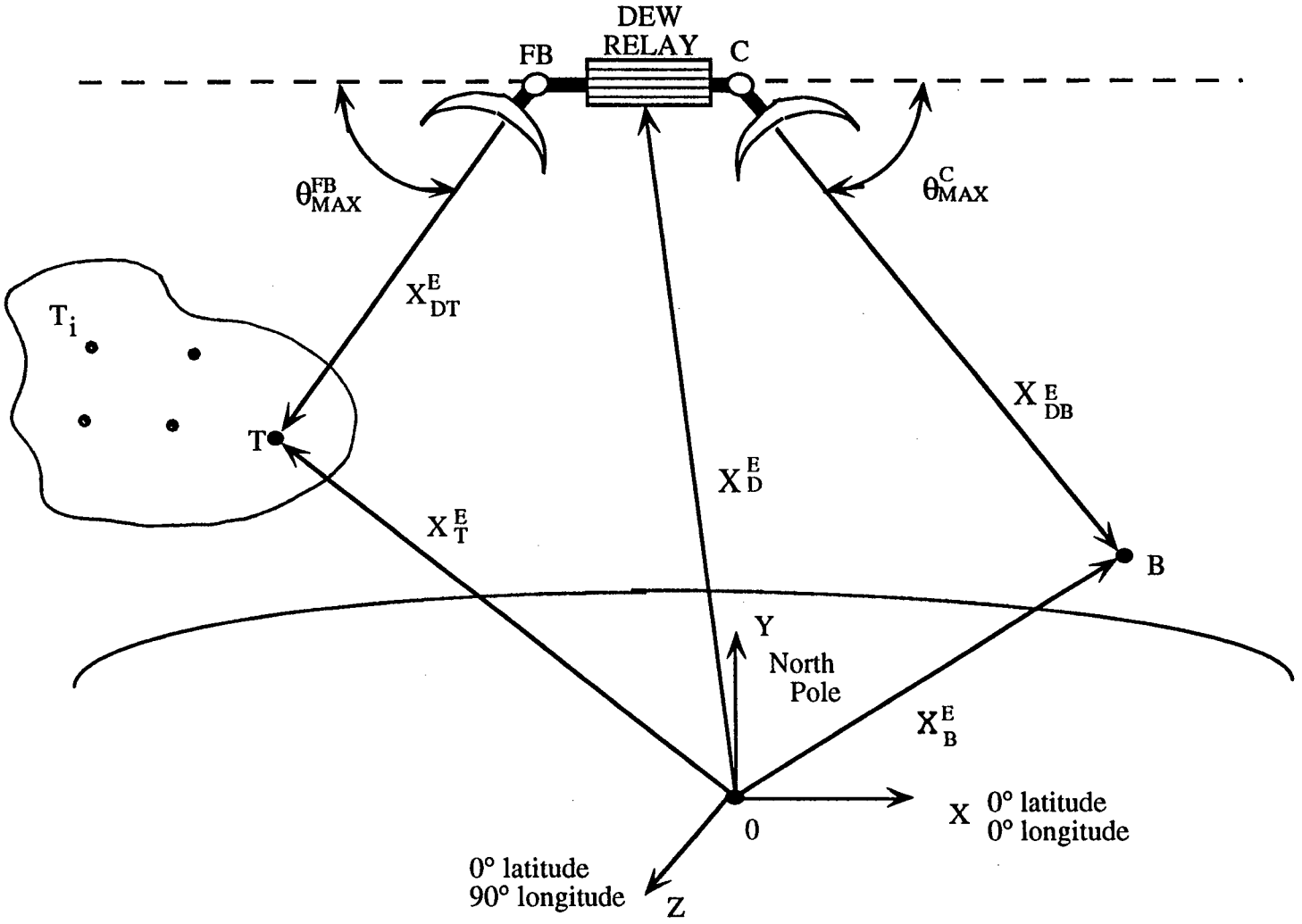

Figure 4.10. Geometry for the ground-based DEW.

The inclusion of this constraint in the expression for retarget time (Eq. (2.21)) yields

$$t^R_{\pi(i)} = t^R_{FS\pi(i)} + t^R_{FB\pi(i)} U\left(\theta_i - \frac{\theta^{FB}_{max}}{2}\right) + t^R_{MB\pi(i)} U\left(\theta_i - \frac{\theta^{FB}_{max}}{2}\right) + k_{GBL} U(A+B) \ , \quad (4.10)$$

where

$$A = \frac{X^E_{DT} \cdot X^E_{DB}}{\| X^E_{DT} \| \| X^E_{DB} \|} \ , \text{ and}$$

$$B = cos\left(\theta^{FB}_{max} - \theta^{C}_{max}\right) .$$

Typically, k_{GBL} will be a large number since considerable time may elapse until a violated constraint is again satisfied.

5. Scheduling Targets During the Midcourse Discrimination Phase

5.1 Introduction

During the midcourse and terminal interception phases, the offense may attempt to conceal its RVs by releasing a mixture of decoys and RV replicas during the post-boost phase. This obviously complicates the threat assessment process and forces the defense to expend additional resources to compensate for the increase in threat uncertainties.

Consider a midcourse threat consisting of a collection $W = \{w_i : i = 1, \ldots, |I|\}$ of $|I|$ *threat clouds* w_i, each cloud having a *density* m_i of *objects* or *targets*, of which r_i are *RV replicas*, d_i are *RV decoys*, and n_i are actual RVs, as shown in Fig. 5.1. While some of the information needed to effectively allocate defensive resources to such a threat

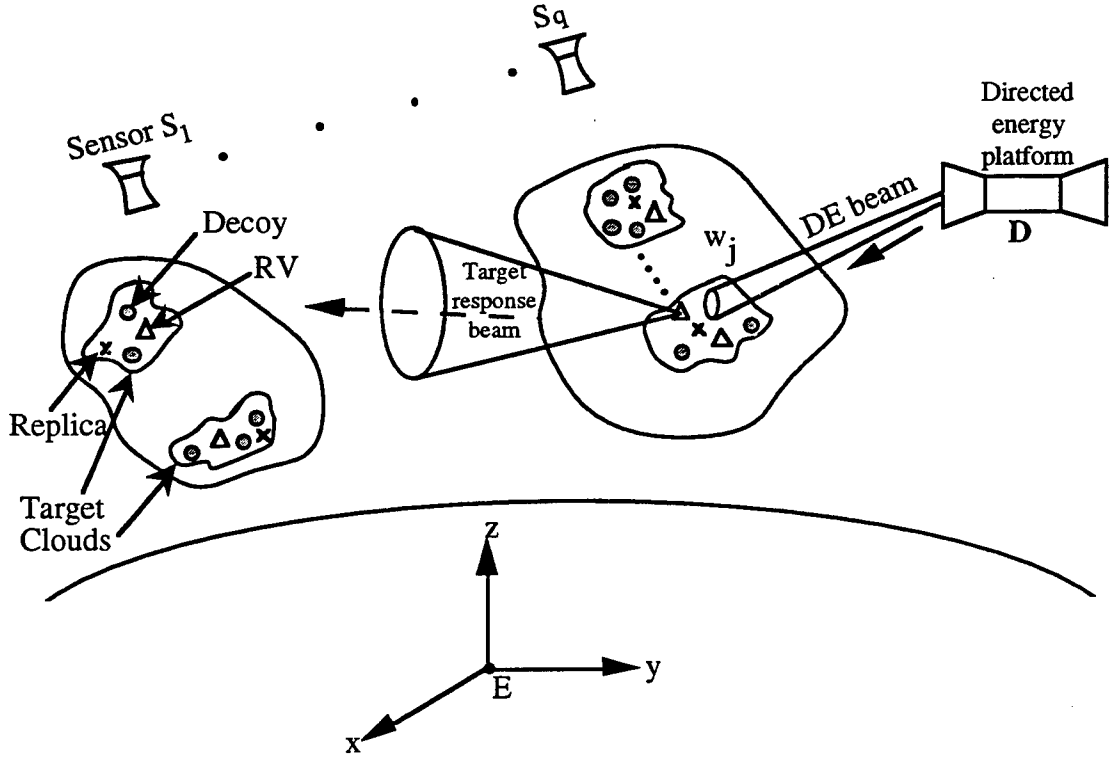

Figure 5.1. In response to an interrogation pulse from the DE platform (D), targets emit a response beam that may be observed by one or more target sensors (S_j) for target classification.

is collected from a variety of sources, the most reliable and timely knowledge about the targets is acquired during the real-time task of *target discrimination*, also referred to as *target classification* or *target identification*.

One way to accomplish this task during midcourse is to deploy a network of directed energy devices and sensors, as in Fig. 5.1, each device designed to *illuminate* or *interrogate* the target in some way, and each sensor stationed to observe and process the results of such an interrogation in order to classify each object as an RV or as a non-RV. A promising concept involves Neutral Particle Beams (NPB) stationed on platforms in low or high earth orbit, one beam per platform, and each platform assigned a subset of targets by a battle manager. Each target T is sequentially illuminated for a duration of τ seconds, regenerating neutral particles whose quantity and energy may be used by the sensor network to decide—or at least to guess—T's type. Although we emphasize the NPB concept in this report, the methods we have developed apply to virtually any active target classification and scheduling problem, and to many passive ones as well.

During this midcourse discrimination phase, the major objective of the platform is to process the collection of targets assigned to it by the battle manager in accordance with a schedule that maximizes overall platform performance. In attempting to achieve this optimality, a precarious tradeoff develops between the statistical confidence in the classification estimates and the time required to reach that confidence. Referring to Fig. 5.2, the risk of misclassifying targets decreases as more time is allocated to the discrimination process, but the risk of missing deadlines and rejecting targets then also increases since the time line is stretched out.

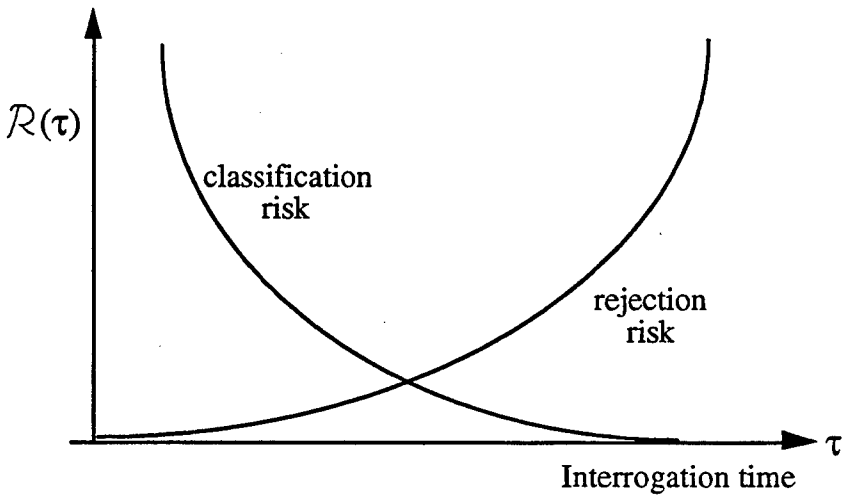

Figure 5.2. As interrogation time τ is increased, classification errors decrease, but less targets can be processed by their due date.

Similar to the boost phase, where leakage probability was our major optimization variable, the interrogation time τ is the most effective control variable during the midcourse discrimination task. But the tour optimization methods developed for the boost phase are needed here as well because in most scenarios retarget times still account for a significant consumption of the allocated time budget.

To gain some insight into our approach, consider Fig. 5.3, where a battle manager hands over a target assignment to the NPB platform. Because the battle manager's knowledge of the threat is incomplete, this assignment contains only generic statistical information:

1. Cloud position and velocity centroids (no individual target state information).
2. Deadlines and release times.
3. Cloud weights and structure.
4. *A-priori* target type information.

This information is refined in real time by on-board tracking and classification algorithms as shown in Fig. 5.3. Following the flow of this figure, and starting with an initial interrogation time of τ_0 seconds, the first step is to estimate the physical response of the target to our NPB pulse of τ seconds, and this requires the modeling of several intermediate physical steps, as shown. The response of a target to an interrogation pulse is observed by the sensor network, and all sensor detections are *fused* into a collective network output. In this report, we employ the very simple fusion policy where the network response equals the response of the sensor whose signal-to-noise ratio is the largest.

The network output is used during the Bayesian update process where prior information about the cloud densities and type distributions are improved into posterior density functions. Together with the losses $L(\theta_i \,|_{\theta_j})$ incurred by classifying a target of type θ_j as one of type θ_i, the posteriors are used to calculate the *expected classification risk* $\mathcal{R}_c(\tau)$ discussed in detail below. Given the interrogation times $\tau = \{\tau_{ij}\}$, the next step is to calculate the *minimal rejection risk* $\mathcal{R}_r(\tau)$, as we did for the boost phase. Recall that this involves finding optimal tours π^* through the threat and, whenever necessary, requires rejecting targets in accordance with a rejection criterion r. While this too will be discussed later, we should mention that, during midcourse discrimination, we employ a two-tier approach to tour construction. In the bottom tier, targets have their usual representation as points or vertices in our graphs, but in the top tier, these vertices represent entire threat *clouds*. Tour optimization is first started at the top tier by finding the best tour connecting cloud centroids. This is followed by finding the shortest tour relating the targets within each cloud. This is obviously an

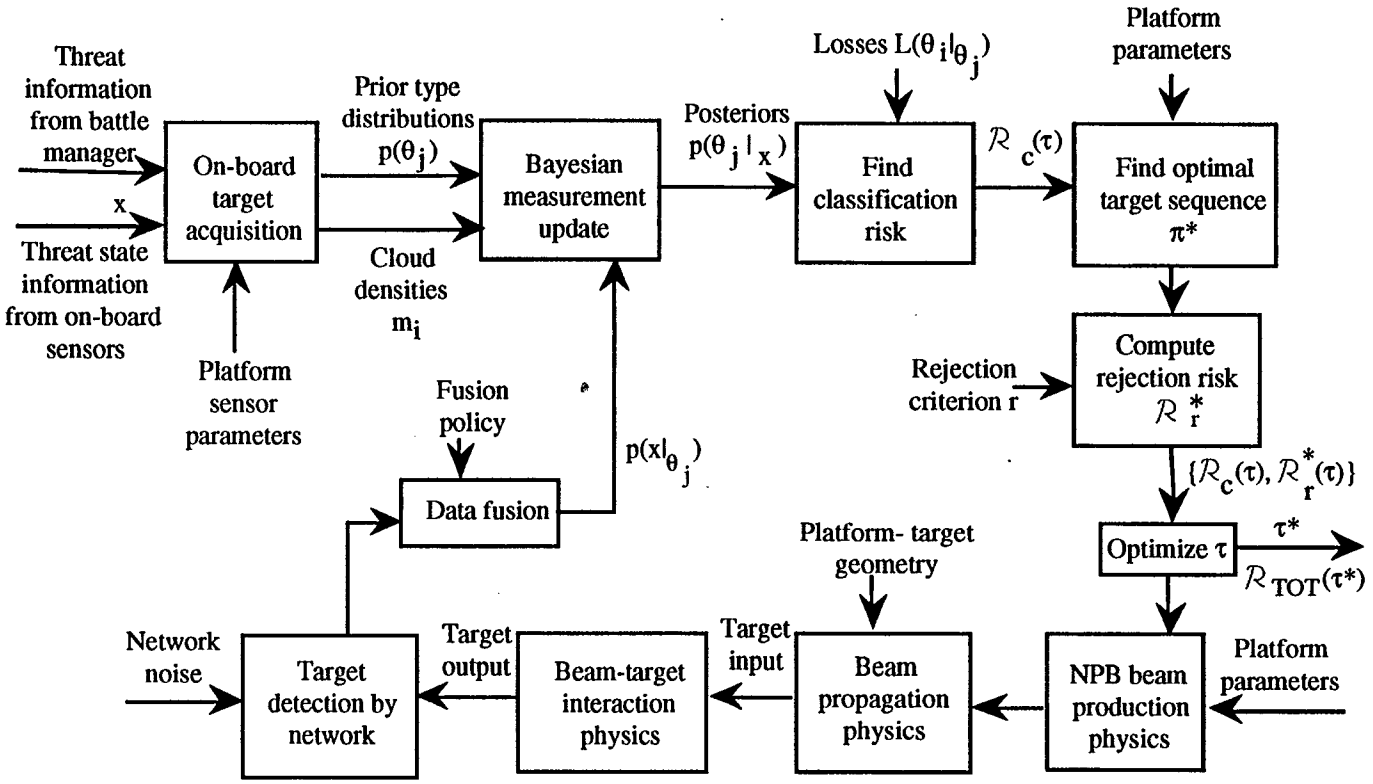

Figure 5.3. The target scheduling problem during the midcourse phase is to (1) find a best tour through the target clouds and (2) allocate an optimal interrogation time to each target.

iterative process since the calculation of cloud tours also requires some estimate of inter-target retarget times, but it results in considerable computational speed-up and usually involves a negligible compromise in overall tour quality.

The total discrimination risk is the sum $\mathcal{R}_{TOT}(\tau) = \mathcal{R}_c(\tau) + \mathcal{R}_r(\tau)$, and this risk is minimized by finding the optimal interrogation time τ_{ij} common to all targets. Similar to equalizing all the p_K's during the boost phase, forcing all τ_{ij}'s to be the same admittedly yields only a suboptimal solution since, in general, each target could require a different interrogation time to achieve global optimality. But our calculations with a similar suboptimal approach in the boost phase have shown that errors resulting from equal τ_{ij}'s are acceptable when computational complexity is considered.

Even though target scheduling is the main focus of this report, it should be clear that no acceptable scheduling solution can be obtained without first solving the target classification problem because targets whose identity is not known cannot be effectively scheduled. Several methods for identifying and classifying objects have evolved over the years,* and these include methods based on Minimax, Bayes risk, and neural or Hopfield nets. In strategic defense problems, the concept of "k-factor" has acquired unargued acceptance in some circles (Holmes and Rocklin 1990; Rocklin and Tolleson 1986).

We rejected minimax approaches because they would lead to highly improbable, non-unique, and time-consuming solutions. We also had several reasons for rejecting neural or Hopfield nets. Such nets do very poorly in time-varying, high-dimensional, and stochastic situations where costs of classification errors may vary, and where considerable modeling robustness is required. Target scheduling also requires not only the effective processing of on-line data, but also the prediction of vehicle states far into the future. In reference to k-factors, we demonstrate below in greater detail that midcourse discrimination problems are strongly *non*-Gaussian, and that costs of classification errors may vary considerably. Furthermore, instead of attempting to meet a predetermined performance criterion that almost surely will be nonoptimal, our scheduling strategy is based on an optimization approach that *assures* the minimization of total risk, at least in principle.

We therefore selected a classical Bayesian risk approach to target classification and we made the following assumptions. All objects in the same cloud have the same deadline and release time, and only one object is illuminated at a time, with a uniform beam that has an abrupt drop-off at its edges (a "cookie-cutter" beam). This simplified model can be improved by introducing an f-spot model associated with a Gaussian beam, as we did in the communication study reported in Corynen and Glaser (1992). If the effects of chaff, loose booster shells, or other forms of intentional or accidental

* Ferguson (1967); Berger (1980); Duda and Hart (1973); Jain (1989); Devijer and Kittler (1987); Joachimsthaler and Stan (1988); Kochler and Evenguc (1990); Seber (1984).

debris turn out to be significant, these can be included by adding another class to the problem.

While we do not explicitly model additional time delays due to: the time-of-flight of neutral particles; their dispersion, processing, counting, and integration; and other waiting times throughout the communication chain, these delays can easily be included in the dwell times allocated to the targets.

This chapter is structured much like Chapter 3. First, in Section 5.2, we introduce the Bayes Risk approach to classification, and we show how it fits into the decision paradigm developed in Chapter 2. Then, in Section 5.3, we address the midcourse target scheduling problem, and we contrast this problem with the boost phase problem by showing how their solutions differ.

5.2 The Target Classification Problem

Coupled with appropriate physical and geometric models, the Bayes Risk approach to classification* is a powerful yet rather straightforward way to obtain optimal object or feature classifications in dynamic environments where measurements are corrupted by various non-Gaussian uncertainties and sources of noise, the costs of classification errors vary considerably, and accurate classifications are needed in real-time.

Before we discuss the statistical situation, let us review the standard *deterministic* pattern recognition problem (Duda and Hart 1973; Jain 1989): Consider a set $\theta = \{\theta_1, \ldots, \theta_c\}$ of c *classes* (also called *states of nature*). In the Target ID Problem, these are the targets that must be classified or identified. Classes in θ are observed or measured via an n-component *feature vector* $x = [x_1, \ldots, x_n]$ ranging over an n-dimensional *feature space* F whose points are called *patterns*. With each class $\theta_i \in \theta$ is associated a subset $F_i \subset F$ such that a feature observation (pattern) x is identified with class θ_i whenever $x \in F_i$. Each class θ_i is typically an *equivalence class* whose points are called *class instantiations*. In some definitions, a vector $[a_1, \ldots, a_c]$ of such instantiations is also called a *pattern*, but we adhere to the first definition in this report.

In the *statistical* pattern recognition problem, classes are similarly defined, although in cases of *unsupervised learning*,† these classes are not well known. We only discuss *supervised learning* in this report, and classes are assumed to constitute a set $\theta = \{\theta_1, \ldots, \theta_c\}$, as before. The measurement process is subject to statistical errors and the features are characterized by conditional probability densities $p(x \mid_{\theta_i})$ on the observations, one density for each class $\theta_i \in \theta$. Upon observing $x \in F$, a more difficult decision must therefore be made in the statistical case to determine the class θ_i associated with this observation. This decision requires the selection of one of m

* Ferguson (1967); Berger (1980); Duda and Haut (1973); Jain (1989); Devijver and Kittler (1987); Lawler (1971a).

† Duda and Hart (1973); Jain (1989); Devijver and Kittler (1987).

possible *actions* from an *action set* $A = \{a_1, \ldots, a_m\}$ in accordance with a *decision function* $d : F \rightarrow A$. This function d is designed to minimize losses resulting from misclassifications.

To find the optimal decision function d^*, let $L(a_i \mid_{\theta_j})$ be the loss incurred whenever action a_i is taken when the true *state of nature* is $\theta_j \in \theta$, and let $p(\theta_j)$ be the a-priori probability of occurrence of class θ_j. When very little is known about the θ_j's, a uniform prior density is used.

The *expected loss* associated with taking decision a_i when x is observed is

$$R(a_i \mid_x) = \sum_{j=1}^{c} L(a_i \mid_{\theta_j}) p(\theta_j \mid_x) \,. \tag{5.1}$$

Considering that every action a_i is determined by the decision function d and the observation $x \in F$, the loss function may be expressed as a function $L(d(x) \mid_{\theta_j})$ of x, and the *conditional risk* associated with observation x is

$$R(x) = \sum_{j=1}^{c} L(d(x) \mid_{\theta_j}) p(\theta_j \mid_x) \,, \tag{5.2}$$

and the overall unconditional classification risk is

$$R = \int_F R(x) p(x) dx) = \int_F \sum_{j=1}^{c} L(d(x) \mid_{\theta_j}) p(\theta_j \mid_x) p(x) dx. \tag{5.3}$$

The *posterior density* $p(\theta_j \mid_x)$ can be computed from the prior density $p(\theta_j)$ and the conditional observation density $p(x \mid_{\theta_j})$ by Bayes' rule:

$$p(\theta_j \mid_x) = \frac{p(x \mid_{\theta_j}) p(\theta_j)}{p(x)} \,, \tag{5.4}$$

where

$$p(x) = \sum_{j=1}^{c} p(x \mid_{\theta_j}) p(\theta_j) \,. \tag{5.5}$$

The density $p(\theta_j \mid_x)$ is called the *Bayes a-posteriori* (posterior) *density*, and the risk

$$R = \int_F \sum_{j=1}^{c} L(d(x) \mid_{\theta_j}) p(\theta_j \mid_x) p(x) dx \tag{5.6}$$

is called the *Bayes Risk.* The decision rule $d^*(x)$ that minimizes this risk is the *Bayes Decision Rule*, and this rule results in the minimal risk

$$R^*(x) = \sum_{j=1}^{c} L(\theta_i \mid_{\theta_j}) p(x \mid_{\theta_j}) p(\theta_j) \; . \tag{5.7}$$

A general sketch of the optimal target classification problem is shown in Fig. 5.4. Although we do not address it explicitly in this report, the *choice* of target classes and measurables strongly influences the quality of classifications. Several criteria are used in arriving at such a choice:

1. Objectives of the classification problem.
2. Target observability and measurability.
3. Economy of parameters used to distinguished one class from another.
4. Discrimination power of the measurables.
5. Computational burden.
6. Overall discrimination optimality.
7. Technological and environmental constraints.

There is general agreement that a mixture of measurements of the radiance, dynamics, and geometry of the targets is needed, but the optimal selection of parameters and measurables is still an active research topic.

The optimal decision rule d^* makes the best use of all the information contained in the feature vectors x whose values represent physical responses and detections of the target interrogation process. Our prior knowledge about physical reality resides essentially in the prior distributions $p(\theta_j)$ and in the conditional distributions $p(x \mid_{\theta_j})$ that model the response of targets whose type is θ_j. This knowledge is extremely varied, including information about the dynamics, basing, targeting, trajectory, and past history of the objects to be discriminated. Targets that may appear to have the same physical characteristics, for instance, can often be identified or discriminated only by knowing their launch points or apparent destinations. Target discrimination thus involves many different kinds of information that have to be *fused* into the $p(\theta_j)$'s and the $p(x \mid_{\theta_j})$'s.

In this report, the NPB return signal is the only physical observable considered for discrimination, and we assume the trivial fusion policy where only the sensor with the highest signal-to-noise ratio is used. But for target identification (Target ID) during the boost phase, plume radiance data must be fused with ATP statistics, dynamics,

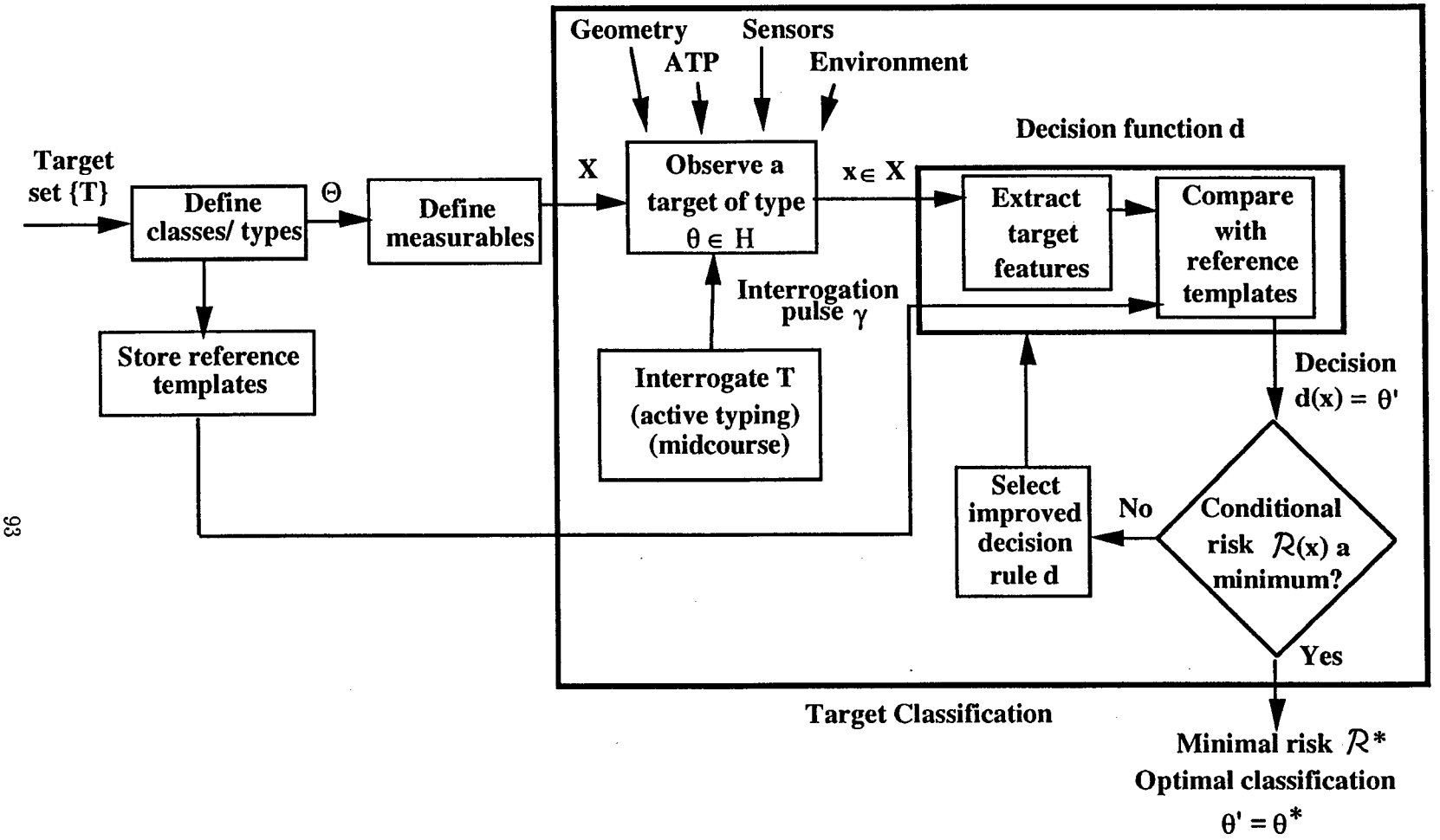

Figure 5.4. Rough sketch of the optimal target classification problem.

basing, and targeting information, and an efficient fusion scheme is absolutely necessary. We are currently addressing this problem, and a detailed treatment of the Bayesian fusion issue is presented in a report currently in progress (Corynen 1993).

Returning to the unconditional risk expression of Eq. (5.6), $d^*(x)$ can be found without integration when it is observed that pointwise minimization of the integrand is sufficient to minimize the entire integral. To find d^*, it thus suffices to find the decision rule that minimizes $R(x)$ of Eq. (5.7) whenever x is observed. This is what is typically done with the Bayesian approach.

It thus suffices to minimize the conditional risk $R(x)$ when x is observed to guarantee that total Bayes risk is minimized, and no integration over the feature space F is required. The optimal target classification rule may therefore be stated as follows:

> If feature value vector x is observed, decide that x originated from a target of type θ_i if, and only if, for all $k \neq i$,

$$\sum_{j=1}^{c} L(d(x) = \theta_i \mid_{\theta_j}) p(x \mid_{\theta_j}) p(\theta_j)) \leq \sum_{j=1}^{c} L(d(x) = \theta_k \mid_{\theta_j}) p(x \mid_{\theta_j}) p(\theta_j)) \qquad (5.8)$$

This is a simple relationship that can effectively be implemented as a real-time algorithm using recently developed methods in discrete optimization.

The target discrimination problem is thus a statistical decision problem that can be described as a four-tuple $TDP =< \mathcal{X}, DM, F, H >$ in accordance with the canonical framework developed in Chapter 2. In the following subsections, we use this description to explain each part of the problem in more precise detail.

5.2.1. The Observation Process $\mathcal{X}$

Two kinds of information are collected during the midcourse discrimination phase: state information about targets, the sensors, and the NPB platform; and target signal return information resulting from target interrogations by the NPB. The first is similar to information collected during the boost phase, and this was discussed in Section 2.2.1.1. The second consists of neutral particle counts received by the sensor network during target interrogation. These counts are not only determined by neutral particle physics, but also by the engagement geometry and the orientation of the vehicles involved: they include noise terms due to space background, other regeneration sources on earth, and detector imperfections. When a given target T is interrogated by an NPB platform D, the overall particle count of any sensor S is therefore a stochastic process:

$$\mathcal{X}(\mathcal{X}_T, \mathcal{X}_D, \mathcal{X}_E, \mathcal{X}_S) = N_T(\mathcal{X}_T, \mathcal{X}_D, \mathcal{X}_S) + N_S(\mathcal{X}_S) + N_E(\mathcal{X}_E)\,, \qquad (5.9)$$

the count due only to the target regeneration of neutral particles from the interrogation pulse, plus the count due only to sensor noise and the noise due only to space and earth background.

In Eq. (5.9), the arguments of the function X have the same meaning as in Section 2.2.1.1.

5.2.1.1. The Target Regeneration Count N_T. Referring to Fig. 5.5, consider a platform D located at X_D, whose neutral particle beam directs a pulse along unit vector e_{DT} towards a target T located at X_T, causing T to regenerate neutral particles, some of which are observed by sensor S located at X_S. In this section, we derive the probability distribution of the particle count N_T detected by S, accounting for the physics of beam generation, propagation, and interaction with the target, and for various acquisition, pointing, and tracking uncertainties.

Assuming a uniform beam of width ψ_D, let I_D be the beam current and τ its pulse length. In some cases, this "cookie cutter" beam profile is inadequate, and Gaussian f-spot models can be introduced, as in Corynen and Glaser (1992), but we ignore such situations. Defining $d_{DT} = \| X_T - X_D \|$, the distance from the NPB platform to the target, the current density at the target is

$$I_T = \frac{4I_D}{\pi\Big(\psi_D d_{DT}\Big)^2} \quad \text{amps/m}^2 \, . \tag{5.10}$$

If the target projects an effective area A_{EFF} for D, the *effective current* received by the target is

$$I_{EFF} = \frac{4A_{EFF}I_D}{\pi\Big(\psi_D d_{DT}\Big)^2} \text{ amps} \, . \tag{5.11}$$

In response to I_{EFF}, the target regenerates ρI_{EFF} neutral particles along e_{DT} with a regeneration beamwidth ψ_T, as shown in Fig. 5.5. At a point d meters downrange along e_{DT}, and assuming a uniform current density inside ψ_T, the current density in vacuum is thus

$$I(d) = \frac{16A_{EFF}I_D\rho}{\pi^2\Big(\psi_D(E)d_{DT}\Big)^2\Big(\psi_T(E)d\Big)^2} \quad \text{amps/m}^2 \, , \tag{5.12}$$

where we recognize that both beamwidths are a function of the particle energy E.

Considering that neutral particle regeneration is a volumetric or bulk mass effect, elongated ellipsoidal targets regenerate more particles when stimulated along their principal axis, and their regeneration angle is smaller along their minor axis. For typical shapes, the target aspect area is modulated by the factor $(1 - \mid 0.5 \cos\theta_{TD} \mid)$, and the regeneration angle by the factor $(0.9 \cos k\theta_{DTS} + 0.1)$,

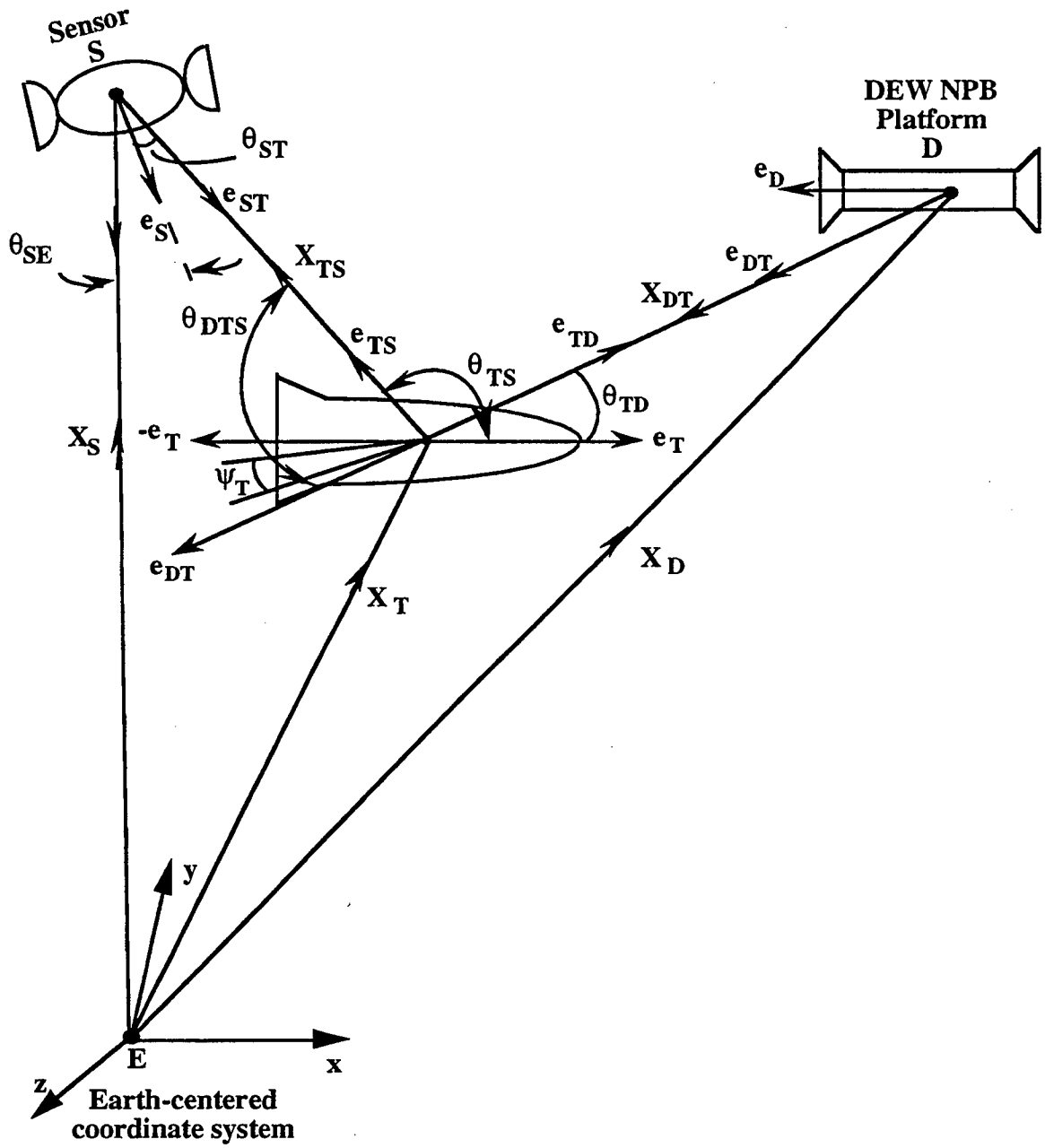

Figure 5.5. Neutral Particle Beam (NPB) target interrogation geometry.

where

$$\cos\theta_{TD} = e_{TD} \cdot e_T.$$
$$\cos\theta_{DTS} = e_{DT} \cdot e_{TS}.$$

$$k = \frac{\pi}{\psi_T}.$$

e_T is a unit vector along the main axis of T.
e_S is a unit vector along the main axis of S.
e_D is a unit vector along the main axis of D.

We next consider path losses due to line-of-sight constraints and due to atmospheric attenuation. Consider Fig. 5.6, where h_0 is the "1/e height" of the atmosphere (h_0 is typically 110 km), and where two satellites are located at X_S and X_T, respectively.

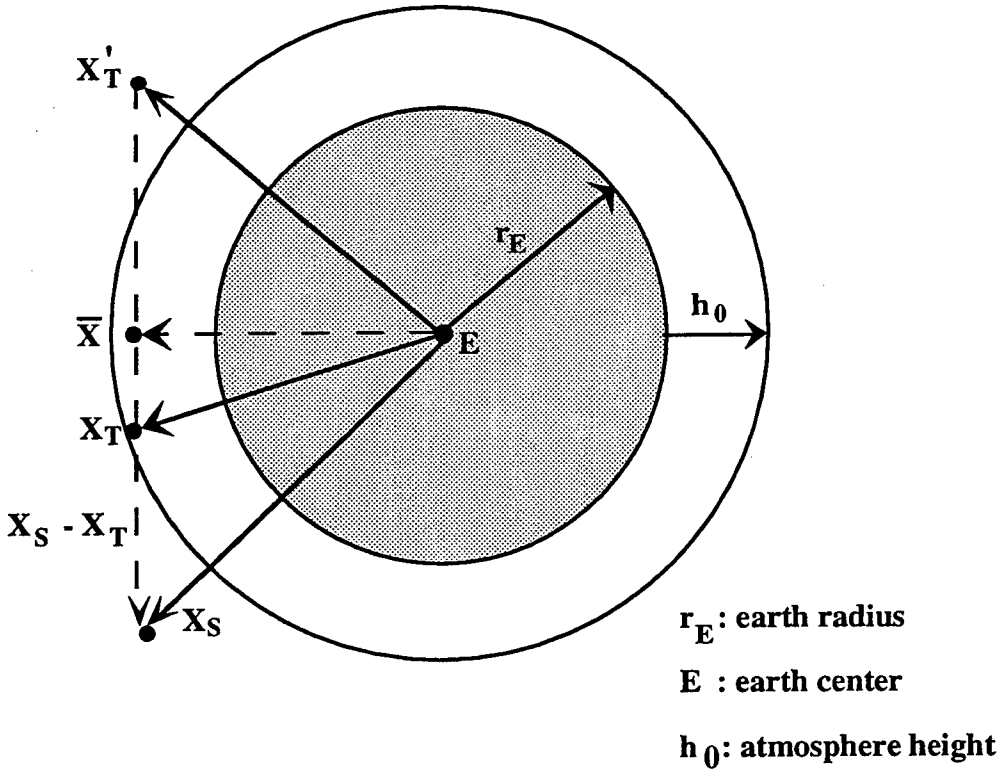

Figure 5.6. Calculating path losses in an exponential atmosphere of height h_0.

To simplify calculations, we assume that the attention factor at an altitude h above the earth surface is

$$\alpha_{ATM} = \left(1 - e^{-h/h_0}\right), \tag{5.13}$$

and that the attenuation experienced by θ beam traveling along an atmospheric path is determined by the lowest point on the path. For the path $X_S - X_T$ in Fig. 5.6,

$$\alpha_{ST} = \left(1 - e^{-h_{min}/h_0}\right),$$

where

$$h_{min} = min\Big\{ \parallel X_T \parallel, \parallel X_S \parallel \Big\} - r_E .$$

This expression is incorrect for paths like $X_S - X_T'$ where the lowest point on the path lies somewhere between X_S and X_T', and may even be inside the earth. If $\overline{X}$ is the lowest point on such a path,

$$\overline{X} = X_T' + \frac{\left(X_S - X_T'\right)\cdot X_T'}{\parallel X_S - X_T' \parallel^2}\left(X_S - X_T'\right). \tag{5.14}$$

The general expression for h_{min} is therefore

$$h_{min} = \begin{cases} max\{0, min\{\parallel X_T \parallel, \parallel X_S \parallel\} - r_E\}, & (X_S - X_T)\cdot X_T < 0 , \\ max\{\overline{X} - r_E, 0\} & , \text{ otherwise} \end{cases} \tag{5.15}$$

The total path loss factor is the product $\alpha_{TOT} = \alpha_{DT}\alpha_{TS}$ of the path loss factors from the NPB platform to the target and from the target to the sensor. Including the cosine angular sensitivity $e_S \cdot e_{ST}$, the effective sensing area A_{SENS}, the particle-to-electron conversion factor ϵ/q, and the total path attenuation factor α_{TOT}, the electron count due to an NPB pulse of τ seconds at a sensor located at a distance of $d_{TS} = \parallel X_S - X_T \parallel$ meters from the target is ($\lfloor x \rceil$ is the nearest integer to X):

$$N_T = \begin{cases} \left\lfloor \dfrac{16 A_{EFF} A_{SENS} I \rho \tau \epsilon \alpha_{TOT} (e_S \cdot e_{ST})}{q\pi^2 (\psi_D(E) d_{DT})^2 (\psi_T(E) d_{TS})^2} \Big[(1 - \mid 0.5 \cos\theta_{TD} \mid)(0.9 \cos k\theta_{DTS} + 0.1\Big] \right\rceil , \\ \qquad \text{for } (-\pi/2k \le \theta_{TS} \le \pi/2k) \bigwedge (e_S \cdot e_{ST} \ge 0) \\ 0 , \qquad \text{for } \Big(\pi/2k \le \theta_{TS} \le -\pi/2k\Big) \bigvee \Big(e_S \cdot e_{ST}\Big) < 0 . \end{cases} \tag{5.16}$$

We shall usually simplify this expression symbolically during further discussions and write

$$N_T = Q\rho\tau A_{EFF} \ , \tag{5.17}$$

where the meaning of all the symbols is obvious.

The effective target area A_{EFF} is derived from the interaction between the NPB and the target using geometric arguments, as follows. Refer to Fig. 5.7, where a target T is shown at a distance d_{DT} from the platform D. The area A_D of the beam at the target is modeled as a circle of radius $r_D = d_{DT}\psi_D/2$, and the effective particle collection area A_{EFF} consists of points both in A_T and in A_D—i.e., $A_{EFF} = A_T \bigcap A_D$, where we have slightly abused the distinction between a "set" and an "area" (area being a *measure* on a set) in order to simplify our exposition.

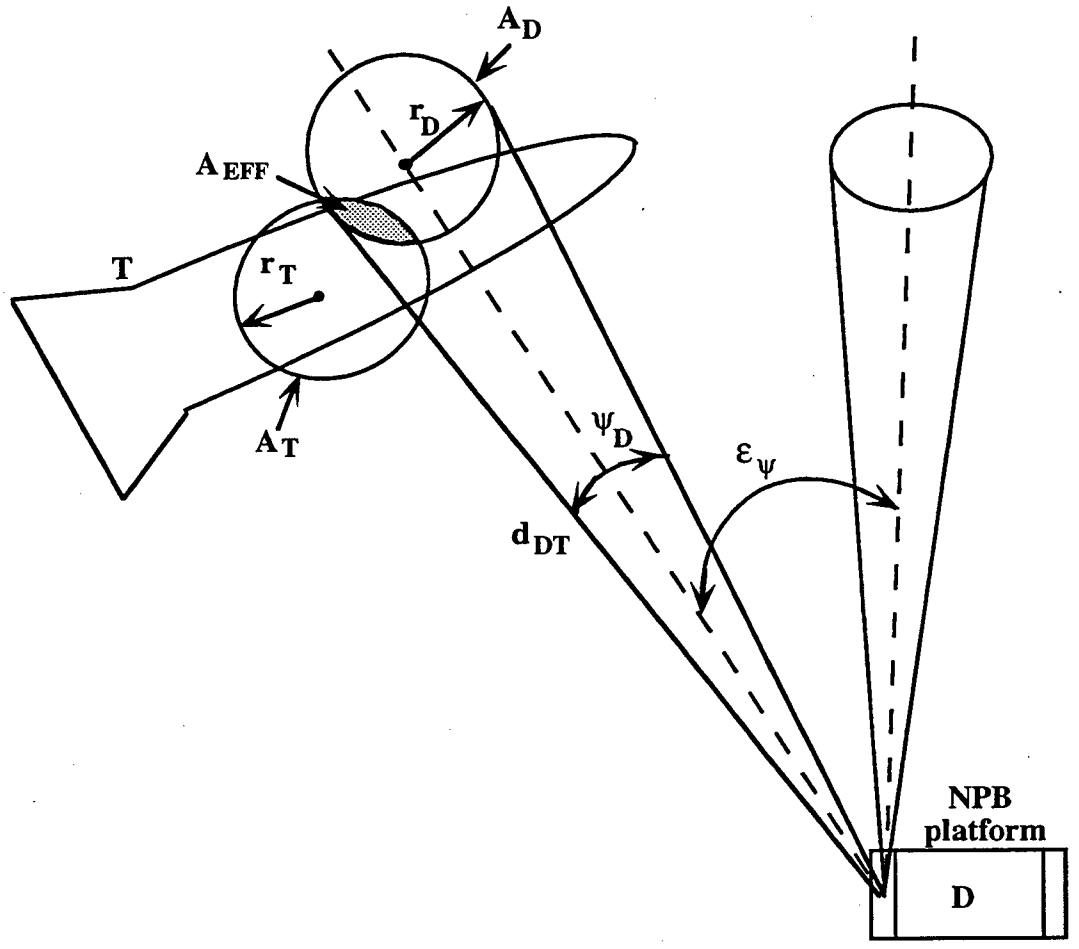

Figure 5.7. Simplified model of NPB-Target interaction where the target (T) is modeled as a projected area A_T of radius r_T.

Two extreme cases are of interest in this analysis. The first is a trivial but also a worst case where no particles are collected by the target because A_T and A_D are disjoint, and $A_{EFF} = A_T \bigcap A_D = 0$. The other case is the best case, where one area is a subset of the other, in which case

$$A_{EFF} = min\Big\{A_T, A_D\Big\} = A^0_{EFF}. \tag{5.18}$$

These two situations are important in practice when acquisition, tracking, and pointing (ATP) errors are considered. When such errors are small, a maximal count N_T may be achieved, but when they are large, N_T may equal zero. Such errors are random variables, of course; thus A_{EFF} is itself a random variable, and so is the particle count N_T.

To derive the distribution of N_T, let ϵ_ψ be the random angular beam direction error in polar coordinates. Then $A_{EFF} = 0$ whenever $\epsilon_\psi > (r_T + r_D)/d_{DT}$ and $A_{EFF} = A^0_{EFF}$ whenever $\epsilon_\psi \leq (\mid r_T - r_D \mid)/d_{DT}$. To compute the probabilities $p_0 = prob(A_{EFF} = 0)$ and $p_1 = prob(A_{EFF} = A^0_{EFF})$, we assume that ϵ_ψ is derived from rectangular errors $\epsilon_{\psi x}$ and $\epsilon_{\psi y}$ whose distribution is Gaussian, with zero means and common variance $\sigma^2_{\psi x} = \sigma^2_{\psi y} = \sigma^2$. Since $\epsilon_\psi = (\epsilon^2_{\psi x} + \epsilon^2_{\psi y})^{1/2}$, the distribution of ϵ_ψ is the Rayleigh distribution (Papoulis 1965), and

$$p_0 = 1 - \mathcal{R}_{\sigma^2}\Big(\frac{r_T + r_D}{d_{DT}}\Big)\,,$$

and

$$p_1 = \mathcal{R}_{\sigma^2}\Big(\frac{\mid r_T - r_d \mid}{d_{DT}}\Big)\,, \tag{5.19}$$

where $\mathcal{R}_{\sigma^2}$ is the CDF of the Rayleigh distribution with variance $0.429\ \sigma^2$ and mean $(\sigma^2\pi/2)^{1/2}$.

Based on the behavior of A_{EFF} as ϵ_ψ is varied, we assume that the CDF of A_{EFF} is increasing linearly from $A_{EFF} = 0^+$ to A^{0-}_{EFF}, as shown in Fig. 5.8, and the mean of A_{EFF} then equals

$$\overline{A}_{EFF} = (1 - p_0 + p_1)\Big(\frac{A^0_{EFF}}{2}\Big)\,. \tag{5.20}$$

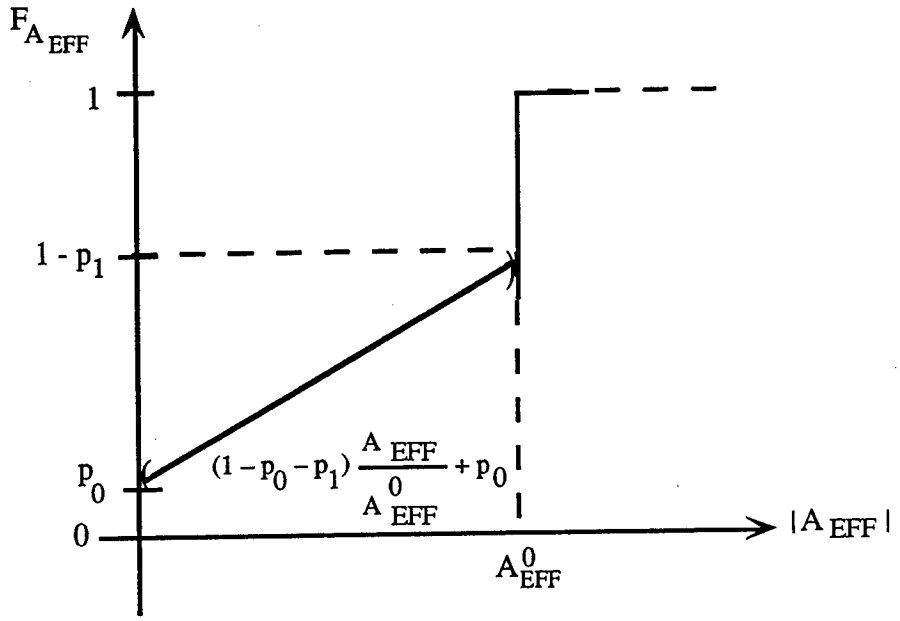

Figure 5.8. The cumulative distribution function of A_{EFF}.

Including the non-random factor $Q\rho\tau$ of Eq. (5.17), and defining $N_T^0 = Q\rho\tau A_{EFF}^0$, the CDF of N_T is obtained via a simple change of variables:

$$F_{N_T}(N_T) = \begin{cases} 0 & , \ N_T < 0 \\ p_0 & , \ N_T = 0 \\ p_1 & , \ N_T = N_T^0 \\ \dfrac{(1-p_1-p_0)N_t}{N_T^0} + p_0 & , \ 0 < N_T < N_T^0 \\ 1 & , \ N_T^0 < N_T \end{cases} \tag{5.21}$$

Using Eq. (5.19), the mean count $\overline{N}_T$ is thus

$$\begin{aligned} \overline{N}_T &= Q\rho\tau\overline{A}_{EFF} \\ &= \frac{(1-p_0+p_1)Q\rho\tau A_{EFF}^0}{2} . \end{aligned} \tag{5.22}$$

5.2.1.2. The Sensor Noise Count N_S. The noise introduced in the counting process by the sensor is due essentially to detector noise, whose statistics are assumed to be Poisson with parameter λ_{DET}. If the noise pulse generation rate is λ_{DET}, the probability that n detector noise pulses are counted during a time interval of size τ is

$$p_{DET}(n;\lambda_{DET}) = \frac{e^{-\tau\lambda_{DET}}(\tau\lambda_{DET})^n}{n!} . \tag{5.23}$$

The mean count $\overline{N}_{DET}$ is $\tau\lambda_{DET}$, and so is the variance. For mean values $\tau\lambda_{DET} > 6$, a Gaussian approximation is accurate and will be used.

5.2.1.3. The Environmental Noise N_E. Earth background noise results from a "porthole view" and is based on maximum albedo flux whose intensity is also Poisson distributed with parameter λ_E, and this noise falls off linearly in θ_{SE} (see Fig. 5.5), as follows:

$$\lambda_E = \begin{cases} \frac{2\lambda_E^0}{\pi}\left(\frac{\pi}{2} - |\theta_{SE}|\right) & , \ 0 \le |\theta_{SE}| \le \pi/2 \\ 0 & , \text{ if not} , \end{cases} , \tag{5.24}$$

where

$$\theta_{SE} = \cos^{-1}\left(\frac{-e_S \cdot X_S}{\| X_S \|}\right) .$$

One additional noise term that usually yields a Poisson count at the sensor is a background effect due to man-made nuclear events, but these are not treated in this report.

5.2.1.4. The Total Count N_{TOT}. The total particle count at a sensor is the sum

$$N_{TOT} = N_T + N_S + N_E , \tag{5.25}$$

whose components were derived above.

Since both N_S and N_E are Poisson distributed, their sum N_{SE} is also, and the random variable N_{TOT} is then simply the sum of two independent random variables, whose distribution may be found by convolution, as in Fig. 5.9, where the densities $f_{N_T}(n)$ and $f_{N_{SE}}(n)$ are shown.

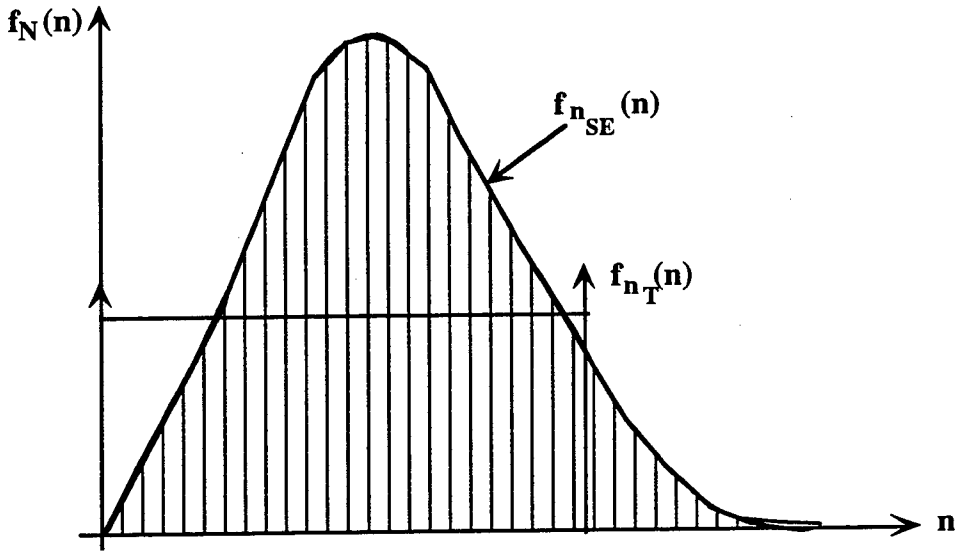

Figure 5.9. The distribution of N_{TOT} is found by convolution.

Recall (Papoulis 1965) that the pdf of the sum $Z = X + Y$ of two independent continuous random variables X and Y is

$$f_Z(z) = \int_{-\infty}^{\infty} f_X(x) f_Y(z - x) dx \, . \tag{5.26}$$

Although neither f_{N_T} nor $f_{N_{SE}}$ are continuous in the conventional sense, their representation as *generalized functions* (Papoulis 1977) is sufficiently smooth to allow the direct application of Eq. (5.26). If we let $Z \equiv N_{TOT}$, $X = N_T$, and $Y = N_{SE}$,

$$f_X(x) = f_{N_T}(n) = p_0 \delta(n) + p_1 \delta(n - N_T^0) + \Big(\frac{1 - p_1}{N_T^0}\Big)(U(n) - U(n - N_T^0)),$$

and

$$f_Y(y) = f_{N_{SE}}(n) = \sum_{i=0}^{\infty} a(i) \delta(n - i) \, ,$$

where:

$$a(i) = \frac{e^{-\tau \lambda_{SE}} (\tau \lambda_{SE})^i}{i!} \, ,$$

and

$$U(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \, . \end{cases} \tag{5.27}$$

To simplify the notation in deriving the CDF of Z, let

$$A_1 = \{X = 0\},\ A_2 = \{X = N_T^0\},\ B_1 = \{Y = 0\},\ B_2 = \{Y = N_T^0\},$$

$$E = \bigwedge_{\substack{i=1\\ j=1}}^{2} \overline{A}_i \wedge \overline{B}_j, \text{ and}$$

$$p_0 = p(A_1),\ p_1 = p(A_2),\ q_0 = p(B_1),\ q_1 = p(B_2).$$

Then

$$F_Z(z) = \sum_{i=j=1}^{2} p\Big(X + Y \le z \mid_{A_i \wedge B_j}\Big) p(A_i \wedge B_j) + p(X + Y \le z \mid_E) p(E)\,, \qquad (5.28)$$

and

$$F_Z(z) = p_0 q_0 U(z) + p_0 q_1 U(z - N_T^0) + p_1 q_0 U(z - N_T^0) + p_1 q_1 U(z - 2N_T^0) + p(E) I(z)\,,$$

where

$$p(E) = \prod_{i=j=1}^{2} \Big(1 - p(A_i) p(B_j)\Big)\,,$$

and

$$I(z) = \int_{-\infty}^{z} \int_{-\infty}^{\infty} \frac{(1 - p_0}{N_t^0} \Big[U(x) - U(x - N_T^0)\Big] \sum_{\substack{i=1\\ i \ne N_T^0}}^{\infty} a(i) \delta(z - x - i) dx dz\,. \qquad (5.29)$$

The integrand of $I(z)$ may be written as

$$g_Z(z) = \int_0^{N_T^0} \frac{(1 - p_0)}{N_T^0} \sum_{\substack{i=1\\ i \ne N_T^0}}^{\infty} a(i) \delta(z - x - i) dx$$

$$= \frac{(1 - p_0)}{N_T^0} \sum_{\substack{i=i^*\\ i \ne N_T^0}}^{i=\lfloor z \rceil} a(i)\,,\ i^* = max\{1,\ \lfloor z \rceil - N_T^0\}\,. \qquad (5.30)$$

In conclusion, the CDF of the total count N_{TOT} is

$$F_{N_{TOT}}(n) = p_0 q_0 U(n) + p_0 q_1 U(n - N_T^0) + p_1 q_0 U(n - N_T^0) + p_1 q_1 U(n - 2N_T^0) +$$

$$p(E) \int_{-\infty}^{n} \frac{(1 - p_0)}{N_T^0} \sum_{i=i^*}^{i=\lfloor z \rceil} a(i) dz\,. \qquad (5.31)$$

5.2.2. The Decision Maker DM

In the target classifiction problem, the decision maker $DM =< d, L >$ classifies observations $x \in \mathcal{X}$ in accordance with a decision function d from the observation space $\mathcal{X}$ to the parameter space Θ, incurring a loss $L\left(\theta_i \mid_{\theta_j}\right)$ when x is classified as originating from a target of type θ_i, whose actual type is θ_j (see Eq. 5.6).

5.2.2.1. The Decision Rule d. Recall that the threat is a collection $W = \{w_i : i = 1, \ldots, \mid I \mid\}$ of $\mid I \mid$ clouds w_i, each consisting of m_i targets of which r_i are RV replicas, d_i are RV decoys, and n_i are actual RVs. Two cases are considered (we omit the subscript i):

Case 1 n, d, and r are known.

Case 2 n, d, and r are not all known.

Case 1

Assuming that RVs generate larger counts than replicas or decoys, the rule is to classify the n largest counts as originating from an RV. More precisely, the observations are m-vectors $(x_1, \ldots, x_m) \in \mathcal{X}^m$, each component count x_j resulting from an interrogation of target T_j in cloud w. The action space is the m-dimensional space $\mathcal{A}^m = \{0,1\}^m$ whose elements are m-tuples $[a_1, \ldots, a_j, \ldots a_m]$, where $a_j = 0$ if the target is classified as an RV and $a_i = 1$ if not. The parameter space Θ is also m-dimensional, and $\Theta = \{(\theta_1, \ldots, \theta_m)\}$, with $\theta_j = 0$, 1, or 2 if the target is an RV, a replica, or a decoy, respectively.

The decision rule for Case 1 is thus a vector function $d : \mathcal{X}^m \rightarrow \mathcal{A}^m$, and $d = [d_1, \ldots, d_k, \ldots, d_m]$, where each $d_k : \mathcal{X} \rightarrow \mathcal{A}$ is a *coordinate* map taking an observation x_k on Target T_k into a decision $d_k(x_k) = a_k \in \mathcal{A} = \{0,1\}$. If we define $d_k(x_k) = 1$ whenever x_k is classified as originating from an RV, and $d_k(x_k) = 0$ when not, our decision rule is

$$d_k(x_k) = \begin{cases} 1, & \text{if } \exists \, integers \, \{r = 1, \ldots, n-m\} \\ & \quad \ni x_{kr} \leq x_k \, , \; k_r \text{ distinct} \, , \quad k_r \neq k \\ 0 \, , & \text{otherwise} \end{cases} \tag{5.32}$$

In Section 5.2.4.4, we show that this rule is optimal.

Case 2

When n, d, and r are all unknown but m is known, the policy is to interrogate and classify targets one at a time using a conventional thresholding approach, as sketched

in Fig. 5.10. The decision function is now the scalar function $d : \mathcal{X} \rightarrow \mathcal{A}$, where $\mathcal{X}$ and $\mathcal{A}$ are as in Case 1, and the rule is to classify a count N_{TOT} as originating from an RV if and only if $N_{TOT} > \eta$.

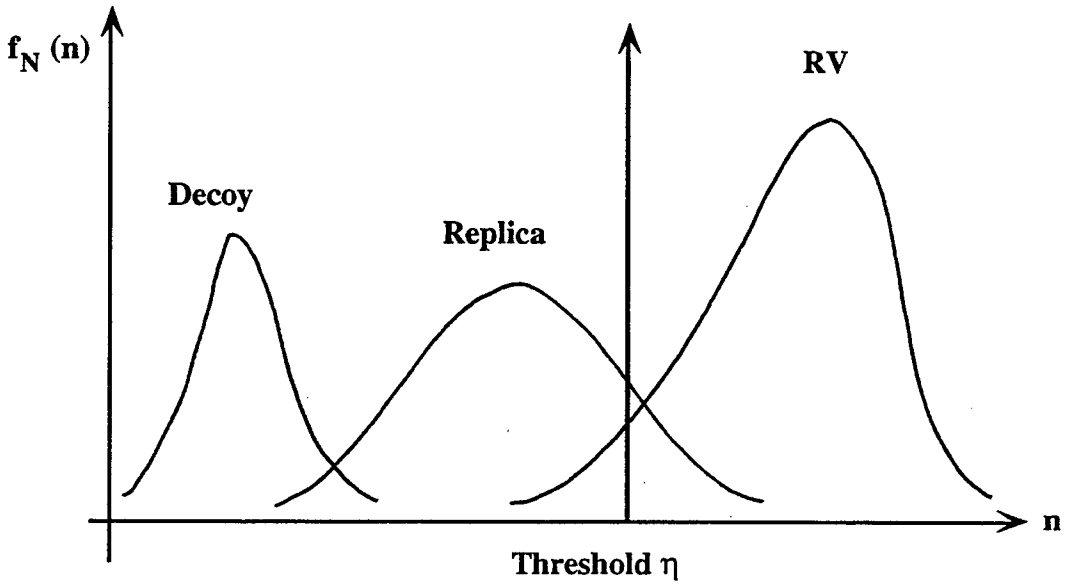

Figure 5.10. The conditional probability densities of the total count $f_{N_{TOT}}(n \mid_\theta)$ for decoys ($\theta = 2$), replicas ($\theta = 1$), and RVs ($\theta = 0$).

5.2.2.2.The Loss Function. As we did in the boost phase, we assume that misclassified or misidentified RVs leak through. We also assume that all RVs in the same cloud have the same value V. In contrast to the more general case where losses are random, even when decisions or actions are known, discrimination losses are fixed once actions have been taken. Again, we consider the two cases defined in Section 5.2.2.1.

Case 1

When n is known, the defense allocates a fixed budget to a cloud, and false alarms do not incur any losses per se. But, for every occurrence of a false alarm (a non-RV is classified as an RV), there must be a misclassification of a non-RV as an RV, since n is fixed. The total cloud loss incurred when taking action $[a] \in \mathcal{A}^m$ when the true state of nature is $[\theta] \in \Theta$ is thus

$$L([a] \mid_{[\theta]}) = V_{RV}\left(\frac{h([a],[\theta])}{2}\right) + nV_{INT}\ , \tag{5.33}$$

where $h([a],[\theta])$ is the *Hamming distance* (Bertsekas and Tsitsiklis 1989) between $[a]$ and $[\theta]$, and V_{RV} and V_{INT} are the costs of leaking an RV and intercepting a target,

respectively. For any pair (x, y) of binary strings of equal length, $h(x, y)$ equals the number of positions (coordinates) where x and y differ.

Case 2

We are only interested in classifying targets as RVs or non-RVs. If $\overline{a}_0$ and $\overline{\theta}_0$ represent the "non-RV" decision and state of nature, respectively, the losses for individual targets are:

$$\begin{aligned} L\Big(a_0 \,|_{\theta_0}\Big) = L\Big(a_0 \,|_{\theta_1}\Big) = L\Big(a_0 \,|_{\theta_2}\Big) = L\Big(a_0 \,|_{\overline{\theta}_0}\Big) = V_{INT}\ , \\ L\Big(\overline{a}_0 \,|_{\theta_1}\Big) = L\Big(\overline{a}_0 \,|_{\theta_2}\Big) = L\Big(\overline{a}_0 \,|_{\overline{\theta}_1}\Big) = 0\ , \text{ and} \\ L\Big(\overline{a}_0 \,|_{\theta_0}\Big) = V_{RV}\ . \end{aligned} \tag{5.34}$$

5.2.3. The Feasibility Set F

The feasibility set is used to specify constraints. Although some deadline constraints exist on the allocation of interrogation times, these are considered in detail during the target scheduling operation which is discussed in the next major section, and no constraints are explicitly imposed on the target identification process.

5.2.4. The Optimization Criterion H

Classification performance is judged by the expected (E) classification cost or loss (L) with respect to the probability measures on observations ($\mathcal{P}_{\mathcal{X}}$). Hence $H = < L, \mathcal{P}_{\mathcal{X}}, E >$.

5.2.4.1. The Loss Function L. The total cloud loss function $L([a]|_{[\theta]})$ for Case 1 is given by Eq. (5.30). For Case 2, losses can only be combined as *risks*, by weighing them by their probability of occurrence, and this is done later in Section 5.2.4.4.

5.2.4.2. The Probability Measure $\mathcal{P}_{\mathcal{X}}$. Different probability functions apply to each of the two cases. In Case 1, a probability measure is needed on the Hamming distances $L([a], [\theta])$ defined earlier. In the thresholding situation of Case 2, more conventional error probabilities apply. In both cases, the prior probabilities on types $[\theta]$ are included by considering $\mathcal{P}_{\mathcal{X}}$ as a collection of conditional probabilities $\mathcal{P}_{\mathcal{X}}\,|_{[\theta]}$, one for each value of $[\theta]$.

Case 1

All that is known about a cloud w is the quantities m, n, r, d, but these are assumed known with certainty. This is therefore a degenerate case of the Bayesian framework (see Eq. (5.6)) where the prior probability $p([\theta]) = 1$ for any combination of types θ_i that satisfies the m, n, r, d requirement, and $p([\theta]) = 0$ otherwise.

In Section 5.2.4.3 below, we show that our decision rule d is optimal and that the only probabilities needed are the probabilities $p(h([a],[\theta]))$ on the Hamming distances $h([a],[\theta])$, and these are derived as follows.

Recall that $N_{TOTRV} = N_{RV} + N_{SE}$, the total count received from an RV, and similarly for N_{REP} and N_{DEC}. The probability p_{RV} that any RV is *correctly* identified is the probability that no decoy signal (N_{DEC}) or no replica signal (N_{REP}) exceeds the RV signal N_{RV}, since the noise term N_{SE} drops out because it is shared by all signal returns. Thus

$$p_{RV} = \Big[p(N_{RV} > N_{DEC})\Big]^d \cdot \Big[p(N_{RV} > N_{REP})\Big]^r . \tag{5.35}$$

The probability that v out of n RVs are *correctly* identified is

$$p_{RV} = \binom{n}{v}(p_{RV})^v(1-p_{RV})^{n-v}, \tag{5.36}$$

and the mean quantity of identified RVs is

$$\mu_{RV} = np_{RV}. \tag{5.37}$$

The probability that $n - v$ out of n RVs are incorrectly identified in thus also $p_{RV}(v)$, and

$$p\Big(h([a],[\theta]) = 2k\Big) = \begin{cases} p_{RV}(n-k), & k = 0,1,\ldots n \\ 0 \quad , & \text{otherwise} \end{cases} . \tag{5.38}$$

The derivation of $p(N_{RV} > N_{DEC})$ and $p(N_{RV} > N_{REP})$ is a bit lengthy. It is reported in Appendix E, where a convolution approach is taken using Eq. (5.32) (also see Fig. 5.8).

Case 2

When no prior information about the parameters θ_j may be assumed, we use a uniform distribution on θ_j. The conditional probability $p(x \mid_{\theta_j})$ for Eqs. (5.4) and (5.6) are simply the conditional count distributions $f_{N_{RV}}(n)$, $f_{N_{REP}}$, and $f_{N_{DEC}}$ derived earlier (see Eq. (5.21)).

The probabilities of incurring the losses of Eq. (5.34) are as follows:

$$p\Big(a_0 \mid_{\theta_0}\Big)(\eta) = p\Big(N_{TOTRV} > \eta\Big) = 1 - F_{N_{TOTRV}}(\eta),$$

$$p\Big(\overline{a}_0 \mid_{\theta_0}\Big)(\eta) = p\Big(N_{TOTRV} \leq \eta\Big) = F_{N_{TOTRV}}(\eta),$$

$$p\Big(a_0 \mid_{\overline{\theta}_0}\Big)(\eta) = p\Big(\Big(N_{TOTREP} > \eta\Big) \vee \Big(N_{TOTDEC} > \eta\Big)\Big),$$

$$= 1 - F_{N_{TOTREP}}(\eta) F_{N_{TOTDEC}}(\eta)\,,$$

$$p\Big(\overline{a}_0 \mid_{\overline{\theta}_0}\Big)(\eta) = p\Big(\Big(N_{TOTREP} \leq \eta\Big) \vee \Big(N_{TOTDEC} > \eta\Big)\Big),$$

$$= 1 - \Big(1 - F_{N_{TOTREP}}(\eta)\Big)\Big(F_{N_{TOTDEC}}(\eta)\Big)\,. \tag{5.39}$$

5.2.4.3. The Objective Function $\mathcal{R}$. Because prior knowledge about threat objects influences the overall classifiction risk, each case has a different objective function.

Case 1

Referring to the general formulation expressed in Eq. (5.6), and omitting the constant interception costs (V_{INT}), the risk for one cloud is

$$\mathcal{R} = \int_{\mathcal{X}} \sum_{\theta \in \Theta} L(d(x) \mid_{\theta}) p(\theta \mid_{x}) p(x) dx\,,$$

$$= \int_{\mathcal{X}} \sum_{\theta \in \Theta} \frac{V_{RV} h(d(x), \theta) p(\theta, x) dx}{2}\,.$$

Exploiting the discrete structure of $h(a, \theta)$, this risk equals

$$\mathcal{R} = \frac{V_{RV}}{2} \sum_{k=0}^{n} 2k p(h(a, \theta) = 2k)\,. \tag{5.40}$$

Using Eqs. (5.36), (5.34), and (E-7),

$$\mathcal{R} n V_{RV} (1 - p_{_{RV}})\,, \tag{5.41}$$

and the total risk for $|I|$ clouds is

$$\mathcal{R}_{TOT} = \sum_{i=1}^{|I|} \mathcal{R}_i\,. \tag{5.42}$$

Case 2

In the thresholding case, targets are classified one at a time, and the cloud risk is the sum of the member risks. The binary nature of d allows the same form as in Case 1. For a single cloud element and for a threshold η,

$$\begin{aligned}\mathcal{R} &= \sum_{i=0}^{1}\sum_{j=0}^{2} L(a_i\,|_{\theta_j})(p(a_i,\theta_j))(\eta)\,,\\ &= \sum_{i=0}^{1}\sum_{j=0}^{2} L(a_i\,|_{\theta_j})(p(a_i\,|_{\theta_j}))(\eta)p(\theta_j)\,,\\ &= V_{INT}\Big[(1-F_{N_{RV}}(\eta))p(\theta_0)+(1-F_{N_{REP}}(\eta)\,F_{N_{DEC}}(\eta))(1-p(\theta_0)\Big]\\ &\quad + V_{RV}F_{N_{RV}}(\eta)p(\theta_0)\,. \end{aligned} \tag{5.43}$$

For a threat consisting of $|I|$ clouds w_i, each of which contains m_i elements T_{ij}, the total classification risk is

$$\mathcal{R}_c = \sum_{i=1}^{|I|}\sum_{j=1}^{m_i}\mathcal{R}_{ij}\,. \tag{5.44}$$

5.2.4.4. Optimizing Classification Risk $\mathcal{R}$. Both cases must be approached differently because, while there exists a closed form solution for Case 1 (the rule of Eq. 5.30), a recursive numerical method must be applied to Case 2.

Case 1

To show optimality of the decision rule for this case, we use the simple fact that signals (N) received from RV's are larger than those from decoys or replicas. Our rule $d^*(x)$ of classifying the strongest signals as originating from RV's implies that,

$$(\forall k \neq 0)\;(p^*(k) = p(h(d^*(x),\theta) = k) < p(h(d(x),\theta) = k) = p(k))$$

for any other rule d. Hence

$$\mathcal{R}(d^*) = \sum_{k=0}^{m} kp^*(k) \leq \sum_{k=0}^{m} kp(k) = \mathcal{R}(d) \tag{5.45}$$

Note the implied assumption that variations in A_{EFF} for different target types cannot reverse the signal dominance of RV's over other targets. But in extreme and rare cases, A_{EFF} could be zero for an RV, and non-zero for a decoy or for a replica, for instance. We ignore such situations.

Case 2

Because closed-form minimization solutions to Eq. (5.43) exist only for the simplest Gaussian cases, and because each cloud w_i will usually have a different optimal threshold η_i^*, a numerical approach is required for this case. A simplified flow of the algorithm is shown in Fig. 5.11, and the meaning of the various block inputs and outputs should be clear from our earlier discussions in this section. But note that our data fusion approach is trival in this first verions of the algorithm, in that we select the output of the sensor that has the highest signal-to-noise ratio $N_T/(N_S + N_E)$ (see Eq. 5.25).

5.3 The Target Scheduling Problem

During the midcourse discrimination phase, the objective is to minimize the *total risk*

$$\mathcal{R}_{TOT} = \mathcal{R}_c + \mathcal{R}_r \ , \tag{5.46}$$

where $\mathcal{R}_c$ is the *classification risk* analyzed in Section 5.2, and $\mathcal{R}_r$ is the *rejection risk* defined in Chapter 2, and analyzed in this section.

As discussed in the introduction to this chapter, both these risks are sensitive functions of the *interrogation dwell time vector* $\tau = [\tau_{ij} : \ i = 1, \ldots, m, \ j = 1, \ldots, m]$, with $\mathcal{R}_c$ decreasing as τ_{ij} is increased and $\mathcal{R}_r$ decreasing as τ_{ij} is decreased, as shown in Fig. 5.2. But, as for the boost phase, $\mathcal{R}_r$ is also a function of the sequence π in which targets are prosecuted, and of the rejection criterion r. In this section, we minimize the rejection risk $\mathcal{R}_r$ by finding optimal values for π, τ, and r, and we combine this minimum risk with the minimal classification risk derived in Section 5.2 to obtain the minimal total midcourse discrimination risk $\mathcal{R}_{TOT}^*$.

5.3.1. Selecting the Permutation π

Exactly the same methods are used to find the optimal target sequence π during the midcourse phase as were used during the boost phase, but they are applied in a slightly different manner. During midcourse discrimination, a two-level hierarchical approach is used in which the nodes or vertices corresponding to the sequence π are target *clouds*, and these are located on the higher level of the hierarchy. During the construction of some tour π, each higher-level node is decomposed into a lower-level tour through the individual targets that constitute the cloud represented by that higher-level node. Retarget times and dwell times are treated similarly, with the total time to complete π equal to the sum of all inter-centroid retarget times and dwell times at each centroid. While these retarget times are treated as they were during the boost phase, the dwell time for any cloud is the sum of the dwell times and retarget times of its constituent member targets, and these are derived from physical and geometrical arguments in a later section. Thus, if t_{uv}^R is the retarget time from cloud w_u to cloud w_v, t_u^D is the dwell time for the cloud w_u, t_{ujk}^D is the dwell time for the j^{th} = target t_{uj} in cloud w_u, and t_{ujk}^R is the retarget time t_{ujk}^R from T_{uj} to T_{uk},

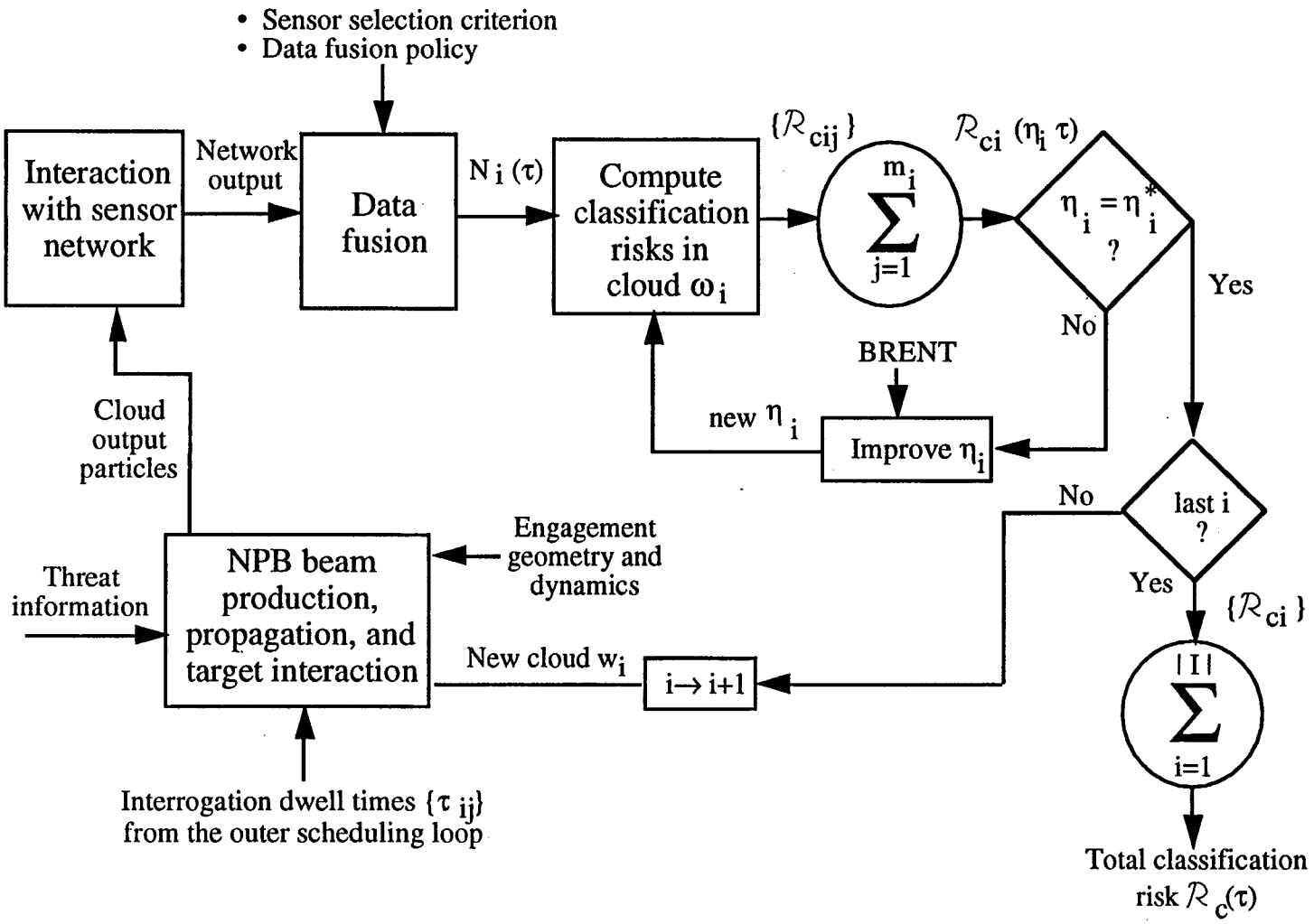

Figure 5.11. In Case 2, target classification risks are minimized by optimizing the decision threshold η_i for each target cloud w_i using the BRENT algorithm.

$$t_u^D = \sum_{j=1}^{m_u} t_{uj}^D + \sum_{\substack{j=1 \\ k=j+1}}^{m_u-1} t_{ujk}^R \,, \tag{5.47}$$

and the total tour time is

$$t_\pi = \sum_{u=1}^{|I|} t_u^D + \sum_{\substack{u=1 \\ v=u+1}}^{|I|-1} t_{uv}^R \,. \tag{5.48}$$

Specific expressions for t_{ujk}^R are derived in the next section, where target rejection is discussed.

5.3.2. Target Rejection

Whenever one or more targets cannot be identified or classified within their allocated time window, some must be ignored, or *rejected.* This process is almost identical to that employed during the boost phase, except for the fact that only target *clouds* are rejected and that the computation of dwell times is different. An individual target is rejected if, and only if, its parent cloud is rejected, and terms in the rejection ratio now refer to clouds instead of targets.

For midcourse discrimination, the rejection ratio for a target cloud w consisting of m targets is

$$r = \frac{\alpha\Big(\sum_{j=1}^{m} V_j - \mathcal{R}_c\Big)}{t^D + \Delta t^R} \,, \tag{5.49}$$

where

α is the cloud deadline hardness,
V_j is the value of target T_j,
$\mathcal{R}_c$ is the cloud classification risk (see Eqs. (5.42–5.44)),
t^D is the cloud dwell time (Eq. (5.47), Section 5.3.1), and
Δt^R is the cloud retarget time rejection gain.

5.3.2.1. Calculating Cloud Dwell Times. Cloud dwell times are sums of target dwell times and retarget times, as Eq. (5.45) shows. Target dwell times are determined in the outer optimization loop, as we discuss in the next section, and every target is allocated the same dwell time during any iteration in that loop. But retargeting times are a bit more complicated since they depend upon the spatial distribution of targets within a cloud. Whereas the rise times and settling times are essentially the same for all targets in a cloud and need only be multiplied by the number of targets in the cloud to obtain a total cloud rise and settling time, the travel time from target to target depends more directly on the cloud geometry and on the method used to sweep through the cloud. We examine two cases.

Case 1: Uniform distribution inside a sphere

We assume that targets are distributed uniformly throughout a spherical cloud of radius r_0, and we compute the probability density of the distance from cloud center to any member target in a projected plane as would be seen on a two-dimensional focal plane array on board the NPB platform. We need to assume, of course, that the clouds are sufficiently far away from the platform to assure that the image on the focal plane array is in fact the mathematical projection of (x, y, z)-points into (x, y)-points.

Starting with a uniform density

$$f_{xyz}(x,y,z) = \begin{cases} \dfrac{3}{4\pi r_o^3}, & x^2+y^2+z^2 \le r_0^2 \\ 0, & \text{otherwise} \end{cases}, \tag{5.50}$$

the marginal density in the x-y plane is

$$f_{xy}(x,y) = \int_{-a}^{a} f_{xyz}(x,y,z)dz = \begin{cases} \dfrac{3(r_0^2-x^2-y^2)}{2\pi r_0^3}, & x^2+y^2 \le r_0^2 \\ 0, & \text{otherwise} \end{cases}, \tag{5.51}$$

where $a = r_0^2 - x^2 - y^2$.

To get the density of $R = (X^2+Y^2)^{1/2}$, consider a differential area $dxdy = rdrd\theta$ in polar coordinates. Then (Fig. 5.12)

$$f_R(r) = \int_0^{2\pi} \frac{3(r_0^2-r^2)^{1/2} r d\theta}{2\pi r_0^3} = \begin{cases} \dfrac{3r(r_0^2-r^2)^{1/2}}{r_0^3}, & 0 \le r \le r_0 \\ 0, & \text{otherwise} \end{cases}, \tag{5.52}$$

and $f_R(r)$ has a maximum at $r = r_0/\sqrt{2}$.

The expected value $\overline{R}$ of R is $3\pi r_o/16 \approx 0.59\ r_0$, and its variance is only $0.04r_0^2$. Considering that the retarget times of objects near the center are relatively small, a good approximation for our purposes is thus to assume that all targets are projected into a circle of radius $0.59r_0$. Including the trip to and from the cloud centroid, the total retarget distance for a cloud is thus $(2 + (0.59)2\pi)r_0$, and the total angular distance travelled is $5.70r_0/d_{DT}\ rad$, where d_{DT} is the platform-to-cloud distance.

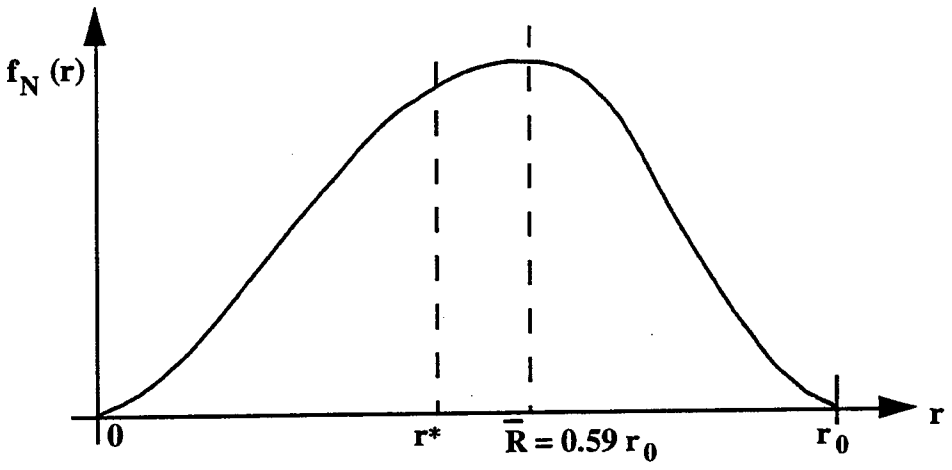

Figure 5.12. Probability density function of the target-to-cloud centroid distance in the projection plane. The expected value of R is $0.59 r_0$, and $f_R(r)$ has a maximum at $r^* = r_0/\sqrt{2}$.

Case 2: Uniform distribution on the surface of a sphere

In spherical coordinates (r, θ, ϕ), targets are distributed uniformly in θ and ϕ, at a constant distance r from the cloud centroid. Since θ and ϕ are independent random variables, their joint density is $f_{\Phi\Theta}(\phi\ , \theta) = 1/2\pi^2$, the product of two uniform densities U_ϕ and U_θ on the intervals $(0, \pi)$ and $(0, 2\pi)$, respectively. The marginal density of Φ is thus simply U_Φ/π, and, using the same assumptions as for Case 1, we may project the spherical distribution onto any plane to obtain the *pdf* of the projected radial variable r, as we did for Case 1. We may thus choose the plane for which $\theta = 0$. Using the conventional transformation to rectangular coordinates (x, y, z),

$$
\begin{aligned}
y =& r_0 \sin\phi \sin\theta\ , \\
x =& r_0 \sin\phi \cos\theta\ , \\
z =& r_0 cos\phi\ , \\
r =& (x^2 + y^2)^{1/2} = r_o(\sin\theta^2 \sin\theta^2 + \sin\theta^2 \cos\theta^2)^{1/2} = r_o \sin\theta\ . \qquad (5.53)
\end{aligned}
$$

Thus $r = r_0 \sin\phi$ (regardless of θ), and the cumulative distribution of the radial distance r is

$$
\begin{aligned}
F_R(r) &= p(R \le r) = p(r_0 \sin\Phi \le r) = p(\sin\Phi \le \frac{r}{r_0}) \\
&= p\left[(\Phi \le \sin_p^{-1}\left(\frac{r}{r_0}\right) \wedge \left(\Phi > \sin_s^{-1}\left(\frac{r}{r_0}\right)\right)\right] \\
&= 2p(0 \le \Phi \le \sin_p^{-1}\left(\frac{r}{r_0}\right) \le \pi/2)
\end{aligned}
$$

$$= \frac{2}{\pi}\left(\sin_p^{-1}\left(\frac{r}{r_0}\right)\right), \tag{5.54}$$

where sin_p^{-1} and sin_s^{-1} are the *primary* and *secondary* inverses of sin, repectively. Thus:

$$f_R(r) = F_R'(r) = \frac{2}{\pi(r_0^2 - r^2)^{1/2}}. \tag{5.55}$$

The expected value of R is $2r_0/\pi$, so, including the start from and return to the cloud centroid, the mean total angular displacement for the cloud is

$$\theta_{TOT} = \frac{6.0\, r_0}{d_{DT}} \text{ rad}. \tag{5.56}$$

The results for these two cases can now be combined with platform slewing parameters to obtained slew times between targets, as we did for the boost phase. These times are simply the sum of the displacement *times* derived in this section (angular displacement divided by the steady state angular slew rate of the platform), and the rise and settling times discussed in Section 2.2.1.3.1.1.

5.3.3 Selecting the Dwell Time Vector $\mathbf{t^D}$

Just as the leakage probability p_L is the most effective scalar control variable during the boost phase, dwell time t^D is the best choice for the midcourse ID phase because ID classification risk $\mathcal{R}_c$ is very sensitive to t^D, and the target rejection risk $\mathcal{R}_r$ is almost exclusively determined by the engagement time line. Optimizing t^D could, with some abuse of language, be referred to as "global optimization," as we did in the boost phase section (Section 4.3), and many comments in that section apply here as well.

As Fig. 5.13 shows, the outer loop of DDTS in the midcourse phase is rather straightforward, given the extensive discussions in previous sections. Total dwell time τ is optimized by minimizing the total risk $\mathcal{R}_{TOT}$, the sum of rejection and classification risk, using the BRENT algorithm, a method for scalar optimization (Brent 1973; Press et al. 1988).

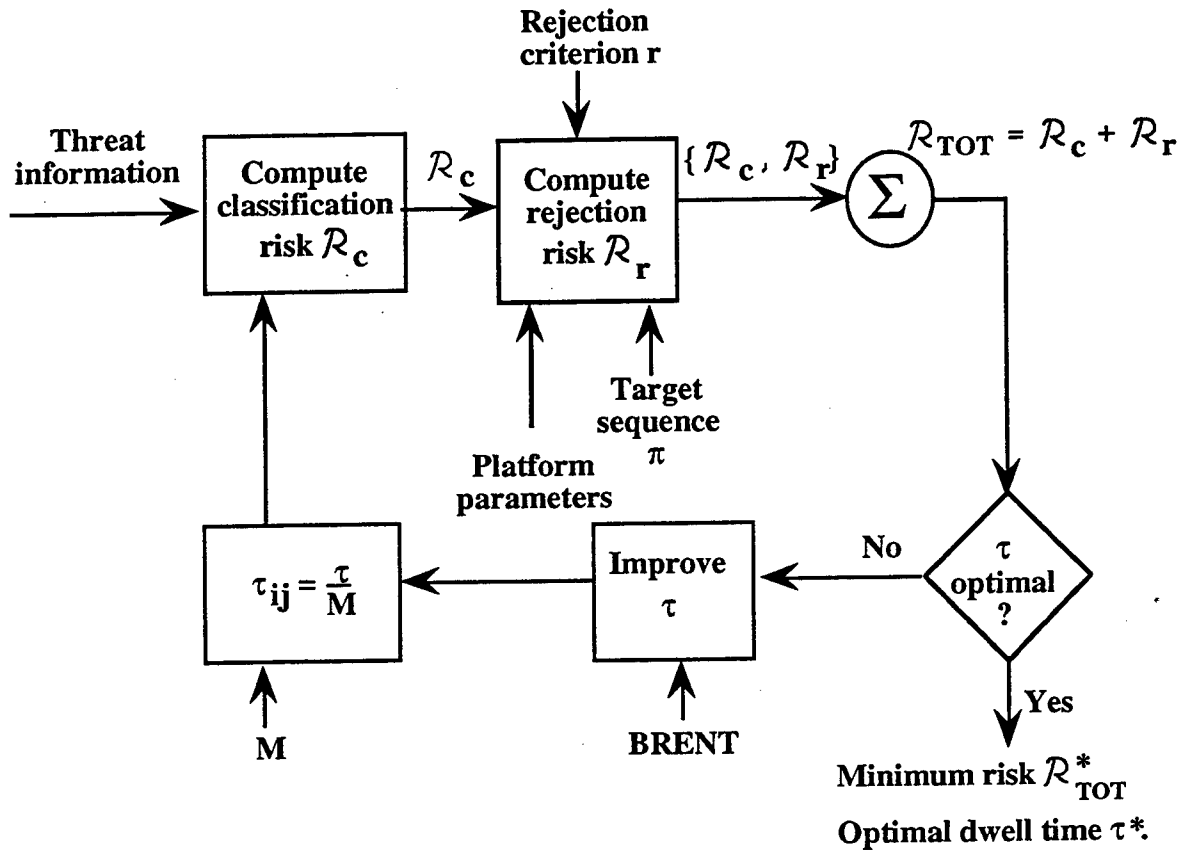

Figure 5.13. The total dwell time τ is optimized in the outer loop and equally divided among all M targets in the threat.

The Brent algorithm is very robust to sharp derivatives, and its application to a variety of problems has always produced rapid convergence, usually in just a few steps. It has also demonstrated surprising forgiveness in addressing problems with multiple local minima, as large as the local "wells" are small compared to that of the global minimum.

6. Testing the DDTS Algorithm

6.1 Introduction

One major project goal was to deliver a target scheduling software package to the Rome Laboratory for incorporation in a simulation testbed and for β-testing towards brass board implemention as a real-time algorithm. The preliminary tests reported in this chapter were conducted with two objectives in mind. First, verification and validation of the internal structure of DDTS was essential to assure that desired parametric relationships were faithfully implemented in the algorithm. Second, experimental evidence was needed to estimate the time and space requirements of the algorithm. Even though a mathematical analysis predicted an expected boost phase complexity of $O(n^3)$, where n is the number of targets processed, additional assurance was sought by exercising the algorithm with a threat driver designed to force the algorithm to reveal its worst behavior. Considering that such a driver was unavailable, the THREATSIM threat simulator was developed on this project to accomplish this driving function. This simulator is described in Section 6.2.

In Section 6.3 we report the results of testing the DDTS algorithm for the boost phase. No midcourse issues were addressed during the testing phase.

6.2 The THREATSIM Simulator

The THREATSIM simulator is currently configured to simulate threats in the boost phase. It is a rather straightforward simulator whose objects (targets) are points with 6 degrees of freedom, capturing both the position and velocity of targets in 3 dimensions. Additional target features include type, value, vulnerability, deadline, release time, ID probability, aspect angle, and deadline hardness.

THREATSIM generates threats in accordance with a specification of two *shape factors*, the *static shape* and the *dynamic shape*. The static shape of a threat defines the geometric distribution of targets at any given time. A linear shape, for instance, means that all targets lie on a straight line, a spherical shape on a sphere, and so on. The dynamic shape specifies how the relative position of targets changes with time. A *divergent* shape, for instance, means that targets are leaving their static shape as time evolves.

THREATSIM has a two-level hierarchical structure where entire target clusters are treated as primitive objects at the higher level, and individual targets are the primitives at the lower level. Our approach is to assign a nominal motion to each target cluster, and to simulate the motion of each individual target in the cluster by adding a disturbance to that nominal motion.

Considering that the principal performance measure for a DEW platform is expected leakage risk, additional parameters must be supplied to DDTS since *risk* is the result of an interaction between targets, sensors, the environment or background, and the DEW platform.

Our discussion is organized as shown in Fig. 6.1.

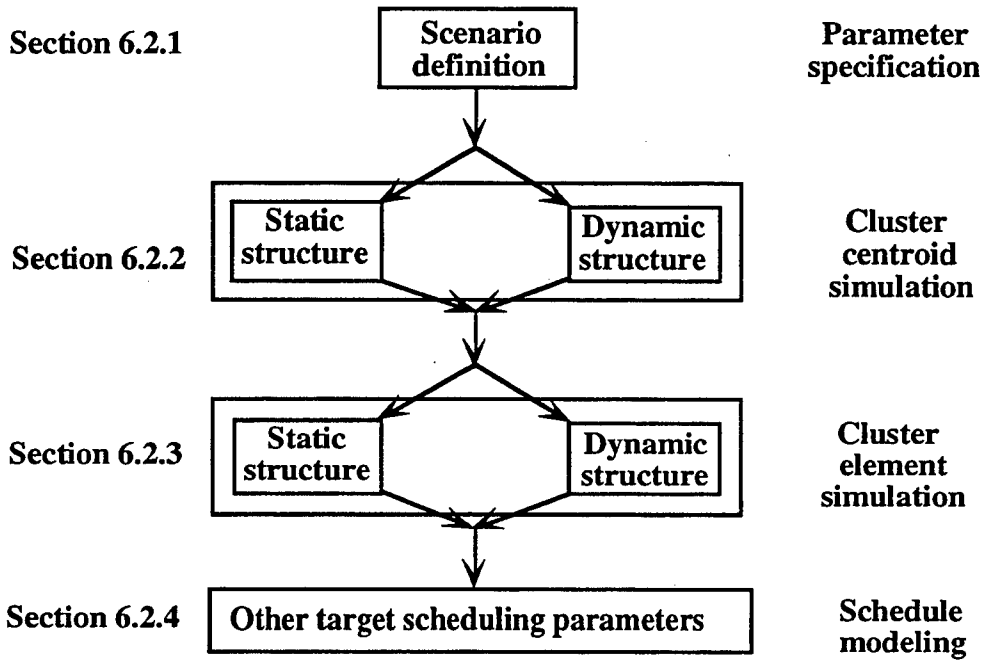

Figure 6.1. Organization of Sections 6.2.1–4.

6.2.1. Scenario and Parameter Specification

For target scheduling purposes, engagements are defined parametrically by specifying a threat, a background and environment, a network of sensors, and a DEW platform. In this section, we list the parameters assigned to each of these major elements; hypothetical values for simulation purposes are included in parentheses, together with their names or labels.

Whereas the simulation of objects in their boost phase is approximated by second-order rectilinear dynamics, we use the conventional characterization of satellite constellations (Corynen and Glaser 1992) discussed below to simulate clusters of targets in the post-boost or midcourse phase. The THREATSIM simulator drives the DDTS algorithm by generating straight-line boost phase segments that converge to a spherical orbit after a controllable amount of time has elapsed.

As shown in Fig. 6.2, our reference coordinate system is earth-centered, with the x-axis containing points with 0 longitude (Greenwich) and 0 latitude. The y-axis has direction 90° longitude and 0° latitude, and the z-axis points to the north pole (0° longitude, 90° latitude).

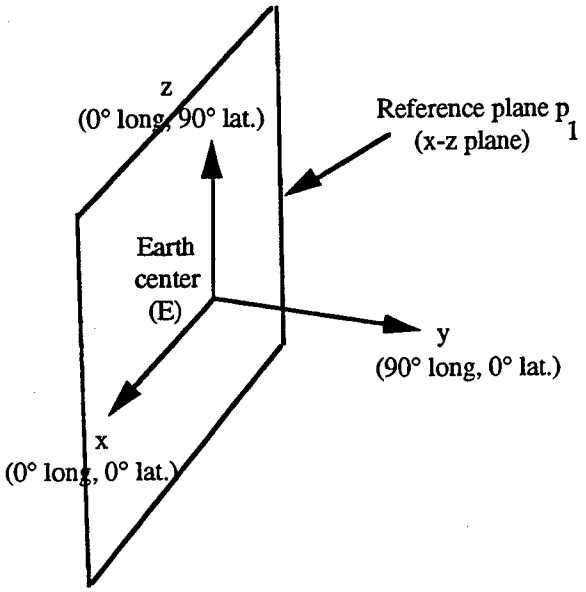

Figure 6.2. Defining the earth-center coordinate system E.

After the boost phase transients have elapsed, targets enter the midcourse, where they are assumed to move in a plane, their *orbital plane*. Such a plane is specified with two rotations of the *reference* or *primal* plane, the x_E–z_E plane in this discussion. The first rotation is a rotation of the primal plane about the z-axis through an angle θ, called the *longitudinal rotation angle*, in the clockwise direction about the z-axis as seen from the point $z = -\infty$ (counterclockwise otherwise). The second is a rotation of the new plane about the x_E-axis through an angle γ, called the *inclination angle* (in conventional coordinate systems, this would actually be the *complement* of the "inclination angle"), in the clockwise direction about the x_{E_1}-axis as seen from the point $x = -\infty$. While the first rotation produces a new coordinate system E_1, the second produces the coordinate system E_2 where satellites move in the x_{E_2}-z_{E_2}-plane (see Fig. 6.3).

The motion of objects in their P_2 plane is specified by an initial angular position in their orbital plane, an offset or *phasing* angle relative to other orbits, and an angular rotation rate w_{E_2} about the origin (E). Both are specified by their orbital radius r.

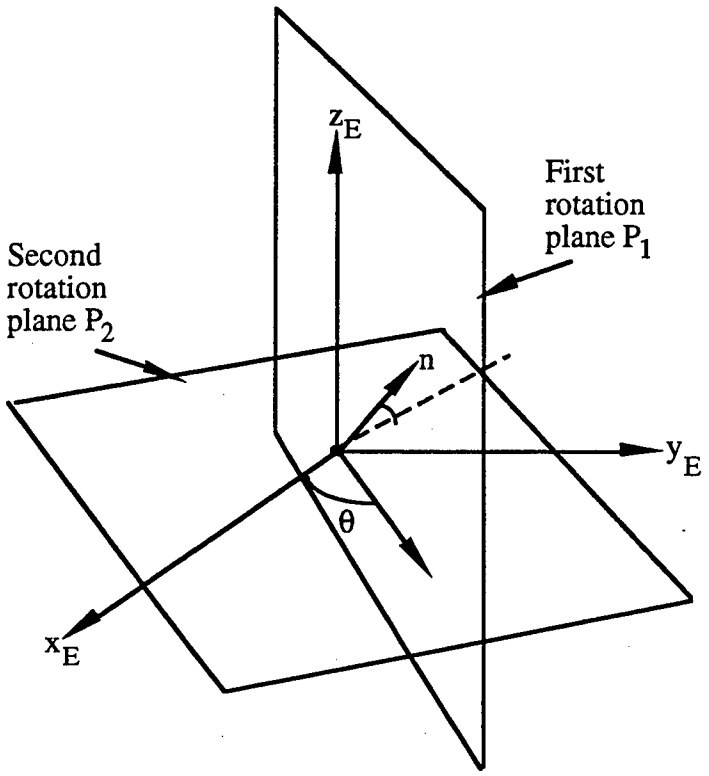

Figure 6.3 Satellite orbital planes p_2 are specified by two sequential rotations, the first about z_E and the second about x_{E_1}.

The angular rates ω along their circle of motion can be derived from elementary physics:

$$\omega = \frac{(GM)^{1.2}}{r^{3/2}} \text{ rad/sec} , \tag{6.1}$$

where (using the MKS system of units)

r $= r_E + h$, the orbit radius of the object (m),
r_E $= 6367.65 \times 10^3$ m, the polar-equatorial average of the earth radius,
h is the orbit altitude above earth surface (m) (typically 400×10^3 m),
M $= 5.9763 \times 10^{24}$ kgm,
G $= 6.67 \times 10^{-11}\ N\text{m}^2(kg)^{-2}$, and
MG $= 2.00 \times 10^7\ kg^{3/2}$.

The rotational (orbital) period is

$$T = \frac{2\pi}{w} \text{ (sec)} , \tag{6.2}$$

and if there are m vehicles in the same orbit, their (uniform) angular spacing (in the x_{E_2}-z_{E_2} plane) is simply

$$\Delta\phi_{k,k+1} = \frac{2\pi}{m} \text{ rad} , \tag{6.3}$$

where k is the counting index of the satellites sharing the same orbit.

To allow an offset or *phasing* between rings, we introduce a *phasing angle* $\phi°$ for each ring, effectively a rotation of the ring about its y_{E_2} axis. By specifying these angles, the first element of ring j is located at an angle $\phi°$ from the x_{E_2} axis (clockwise rotation about the y_{E_2} axis as seen from $y = +\infty$). Since the rings are well-ordered in accordance with longitudinal rotation, the distribution of targets in each ring is fully specified.

For a given ring of density m_j, therefore, the direction angle (in the x_j–z_j plane) to the initial (t = 0) position of target k_j in the ring is simply

$$\phi_{kj} = \frac{(k_j - 1)2\pi}{m_j} + \phi_j^\circ , \quad k = 1, \ldots, m_j.$$

Summarizing, the motion of an object k is described by a vector in the E_2 system:

$$X_{E_{2,k}}(t) = (x_{E_{2,k}}(t),\ y_{E_{2,k}}(t),\ z_{E_{2,k}}(t))^T , \tag{6.4}$$

where

$x_{E_{2,k}}(t) = r \cos \alpha_{E_{2,k}}(t)$,
$z_{E_{2,k}}(t) = r \sin \alpha_{E_{2,k}}(t)$,
$y_{E_{2,k}}(t) = 0$,
$\alpha_{E_2}(t) = \omega_{E_2} t + \phi_k + \phi°$, and
ω_{E_2} is the angular rotation rate given by Eq. (6.1).

To express $X_{E_2}(t)$ in the primal earth-centered system E,

$$X_E(t) = Rot_z^{-1} Rot_x^{-1} X_{E_2}(t) , \tag{6.5}$$

where

$$Rot_z^{-1} = Rot_z^T, \text{ and } Rot_z = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} , \tag{6.6}$$

in which θ is the longitudinal rotation angle,

$$Rot_x^{-1} = Rot_x^T, \text{ and } Rot_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & \sin\gamma \\ 0 & -\sin\gamma & \cos\gamma \end{bmatrix} , \tag{6.7}$$

in which γ is the inclination angle, and $X_{E_2}(t)$ is a column vector.

Conversely, to express a vector X_E in the new coordinate system E_2,

$$X_{E_2} = Rot_x Rot_\gamma X_E \ . \tag{6.8}$$

Now that we know how to specify the motion of individual targets, a *constellation* of targets is specified by five parameters:

- An inclination angle (γ).
- A quantity of rings (n_r).
- A set of ring elevations $\{r_j,\ j = 1,\ \ldots,\ n_R\}$ (maximum altitude above earth).
- A set of ring sizes $\{n_j,\ j = 1,\ \ldots,\ n_R\}$ (quantity of satellites in each ring).
- A set of phase angles $\{\phi_j^\circ,\ j = 1,\ \ldots,\ n_R\}$.

For each constellation, the inclination angle (γ) is fixed and the quantity n_R of rings determines the angular separation $\Delta\theta$ in the x_{E_2}-y_{E_2} plane (longitudinal rotation angle). Clearly,

$$\Delta\theta = \frac{2\pi}{n_R} \text{ rad} \ . \tag{6.9}$$

Next, the ring elevations r_j determine the motion of each object in the ring, and the ring sizes n_j specify the angular spacing $(\Delta\theta_{i,i+1})_j = 2\pi/n_j)$ between objects in ring j. Conventionally, any object in a given ring is identified by a *position number* k (a "counting index"). This number is determined by the first object in the ring and the quantity of objects preceding the given object in the x_{E_2}-z_{E_2} plane as shown in Eq. (6.4).

Another important coordinate transformation problem is to transform vectors in the earth-centered system E to and from the rotating platform coordinate systems E_R and E_T (we shall simply use E_P in this discussion). To start, we assume that the negative z-axis of every target goes through earth center, and that its y-axis is aligned with its velocity vector. Given the platform's position in system E, its coordinate system is thus fully specified, as follows.

In the E_2 system defined by the orbital plane of a platform, the position of the k^{th} platform is

$$X_{E_2}^P(t) = r\cos(\omega t + \phi_k + \phi^\circ)\ \mathbf{e}_{E_2x} + r\sin(\omega t + \phi_k + \phi^\circ)\mathbf{e}_{E_2z} \ , \tag{6.10}$$

where r and ω were defined earlier.

The velocity is

$$\dot{X}^P_{E_2}(t) = \frac{dX^P_{E_2}(t)}{dt} = \omega r \Big[-(\cos(\phi_k + \phi^\circ) \sin\ wt + \sin(\phi_k + \phi^\circ) \cos\ \omega t)\mathbf{e}_{E_{2x}} + (\cos(\phi_k + \phi^\circ) \cos\ \omega t - \sin(\phi_k + \phi^\circ) \sin\ \omega t)\mathbf{e}_{E_{2z}} \Big] \quad (6.11)$$

Given the orientation of its z-axis and y-axis in the E_2 system, the unit vectors specifying the platform's coordinate system are the unit basis vectors

$$\mathbf{e}^P_{E_{2z}}(t) = \frac{X^P_{E_2}(t)}{\| X^P_{E_2}(t) \|} ,$$

$$\mathbf{e}^P_{E_{2y}}(t) = \frac{\dot{X}^P_{E_2}(t)}{\| \dot{X}^P_{E_2}(t) \|} , \quad (6.12)$$

$$\mathbf{e}^P_{E_{2x}}(t) = \mathbf{e}^P_{E_{2y}}(t) \times \mathbf{e}^P_{E_{2z}}(t) .$$

Expressed in system E, whose unit basis vectors are $\mathbf{e}_{Ex}$, $\mathbf{e}_{Ey}$, $\mathbf{e}_{Ez}$, and omitting time t for convenience,

$$\mathbf{e}^P_{E_{2x}} = Rot_z^{-1} Rot_x^{-1} \mathbf{e}^P_{E_{2x}} ,$$

$$= p_{11}\mathbf{e}_{Ex} + p_{12}\mathbf{e}_{Ey} + p_{13}\mathbf{e}_{Ez} .$$

Similarly,

$$\mathbf{e}^P_{E_{2y}} = p_{21}\mathbf{e}_{Ex} + p_{22}\mathbf{e}_{Ey} + p_{23}\mathbf{e}_{Ez} ,$$

and

$$\mathbf{e}^P_{E_{2z}} = p_{31}\mathbf{e}_{Ex} + p_{32}\mathbf{e}_{Ey} + p_{33}\mathbf{e}_{Ez} . \quad (6.13)$$

Now let $\mathbf{X}_p$ be some vector in the platform system E^P_2. If $\mathbf{X}_E$ is the same vector expressed in the coordinate system E, then

$$\mathbf{X}_E = P^T \mathbf{X}_p + Rot_z^{-1} Rot_x^{-1} \mathbf{X}^P_{E_2} , \quad (6.14)$$

where $P = [p_{ij}]$, the matrix of coefficients p_{ij} defined above and located in row i and column j, and $\mathbf{X}^P_{E_2}$ is the position of the platform. Hence,

$$X_p = P[\mathbf{X}_E - Rot_z^{-1} Rot_x^{-1} \mathbf{X}^P_{E_2}] , \quad (6.15)$$

and P provides the rotation required to express any vector in E as a vector in the rotating platform coordinate system E_2^P.

6.2.1.1. The Threat. Threats are defined by specifying global characteristics and individual target characteristics.

6.2.1.1.1. Global Threat Characteristics. These are specified with the following parameters:

1. Total quantity of targets (N=100).
2. Quantity and size of target clusters ($M_i = 10$) (Clouds in the midcourse).
3. Start time for launch ($t_o = 74$ seconds).
4. Threat shape (discussed below).
5. Shape of threat dynamics (discussed below).
6. Cluster centroid position and motion ($\widehat{X}$, $\dot{\widehat{X}}$).
7. Cluster initial position and motion ($\widehat{X}_0 = (-5000, 0, 0)$, $\dot{\widehat{X}} = (1, 1, 1)\ km/sec$).

6.2.1.1.2. Individual Target Characteristics. Every target is described by a multidimensional vector whose components are:

1. Position vector in 3 dimensions (X).
2. Velocity in 3 dimensions ($\dot{X}$, expected cluster perturbation of 1 km/sec).
3. Deadline and release time (t_D, t_R).
4. Dwell time (t^D, computed on line).
5. Leakage loss/value ($V = 1$).
6. Energy consumed in response to DEW illumination (E).
7. Probability of identification ($P_{ID} = 1$, except when used as independent variable).
8. Type (θ is the same for all targets).
9. Deadline hardness ($\alpha_D = 1$ for all targets).
10. Vulnerability radius (r_T or r_{AIM}, 1 meter).
11. Aspect angle (computed from (1) and (2)) (α_T).

12. Hardness mean and variance ($\mu_H = 10^5 J/cm^2$, $\sigma_H = 10^4 J/cm^2$).

13. Initial position and velocity ($X_0, \dot{X}_0$).

6.2.1.2. Background and Environment (B&E). During the boost phase, B&E noise and disturbances are included as statistical noise terms in the sensor and DEW platform ATP parameters. During midcourse, additional Poisson noise terms are added to the neutral particle detection signals at the sensors, and atmospheric attenuation factors are also included.

6.2.1.3. The Sensor Network. The data fusion policy in this report consists of choosing the sensor whose signal is largest. Therefore, only one sensor is considered herein, and its parameters are:

1. Sensor position (X_S) and orientation (β_S).

2. Sensor velocity ($\dot{X}_S$) and rotation rate (ω_S).

3. Discrimination matrix (its entries are probabilities p_{ij} and costs c_{ij} of classifying Target i as Target j).

4. Sensor beamwidth and total field of view (FOV).

5. Tracking bias and uncertainties.

6. Initial position and velocity ($X_{S0}, \dot{X}_{S0}$).

6.2.1.4. The DEW Platform. The DEW platform performs various operations that are critical to target scheduling. The platform consists of three major hardware components: the main platform (P), the forebody (FB), and the fast steering head (FS). Each of these components and functions are characterized by several parameters:

1. DEW beamwidth (ϕ_D or $\theta_{DEW} = 0.15\mu rad$).

2. Wavelength ($\lambda = 3.5\mu m$).

3. DEW and platform maneuvering (thrust) power (P_D and P_T).

4. Bias and jitter ($b_\theta = 10^{-7} rad, \epsilon_\theta \approx N(0, 10^{-7} rad)$).

5. Energy consumed and energy limit (E and E_{max}).

6. Platform tracking point bias and jitter (ϵ_T, σ_T).

7. DEW efficiency ($\eta_{DEW} = 0.9$).

8. Maximum angular displacement ($U^{FB}_{max} = 1$ rad, $U^{FS}_{max} = 10^{-3}$ rad, $U^{P}_{max} = \pi$ rad).

9. Displacement input saturation level ($U_{SAT}^{FS} = 10^{-3}$ rad, $U_{SAT}^{FB} = 10^{-1}$ rad, $U_{SAT}^{P} = 1$ rad).

10. Maximum slew rate ($\dot{c}_{max}^{FS} = 10$ rad/sec, $\dot{c}_{max}^{FB} = 1$ rad/sec, $\dot{c}_{max}^{P} = 0.1$ rad/sec).

11. Pointing variance ($\epsilon^{FS} = 10^{-7}$ rad, $\epsilon^{FB} = 10^{-3}$ rad, $\epsilon^{P} = 10^{-1}$ rad).

12. Damping constant ($\delta^{FS} = \delta^{FB} = \delta^{P} = 0.5$).

13. Critical frequency ($\omega_n^{FS} = 10^4$ rad/sec, $\omega_n^{FB} = 10^3$ rad/sec, $\omega_n^{P} = 10$ rad/sec).

14. Position and velocity vector $(X, \dot{X})$.

15. Initial position and velocity vector ($X_0 = (10^3, 10^4, 0)$ km, $\dot{X}_0 = (-10, 10, 0)$ km/sec).

16. Optics diameter ($D = 8 - 13$m).

17. Optics power ($P = 1.5 \times 10^7$w/ster.).

18. Propagation attenuation constant ($k_T = 10^{14}$m^2).

6.2.2. Simulating Cluster Centroid Motion

The motion of clusters is simulated by randomly selecting their centroid position and velocity to achieve various shapes or "profiles".

6.2.2.1. Cluster Centroid Position. We have k *target clusters* $C = \{c_1, \ldots c_i, \ldots c_k\}$ whose position centroids $\widehat{X} = \{\widehat{X}_1, \ldots, \widehat{X}_i, \ldots \widehat{X}_k\}$ are randomly selected to obtain combinations of three basic shapes: linear, spherical, or random. Usually, the position "profile" thus obtained consists of a linear segment during boost phase followed by a circular segment reached asymptotically during midcourse.

6.2.2.1.1. Linear Profile. For this profile, the centroid position $\widehat{X}_i$ of cluster C_i is generated by the recursive law

$$\widehat{X}_i = \widehat{X}_{i-1} + \widehat{a}_i \widehat{V}_i \,, \tag{6.16}$$

where

$\widehat{X}_1$ is the specified position of the first cluster, and
$\widehat{a}_i \approx U[\alpha_a, \alpha_b]$, the uniform distribution in $[\alpha_a, \alpha_b]$, $-\infty \leq \alpha_a < \alpha_b \leq \infty$.

The vector $\widehat{V}_i$ is the *direction vector* for C_i whose components $\widehat{V}_{ix}$, $\widehat{V}_{iy}$, and $\widehat{V}_{iz}$ have a uniform distribution in the intervals $0 \leq \widehat{V}_{xa} < \widehat{V}_{xb}$, $0 \leq \widehat{V}_{ya} < \widehat{V}_{yb}$, $0 \leq \widehat{V}_{za} < \widehat{V}_{zb}$. The *mean direction vector* is

$$\bar{\widehat{V}} = \left(\frac{\widehat{V}_{xb} - \widehat{V}_{xa}}{2}, \frac{\widehat{V}_{yb} - \widehat{V}_{ya}}{2}, \frac{\widehat{V}_{zb} - \widehat{V}_{za}}{2}\right),$$

and simulations are typically designed so that V lies along a vector from the cluster launch point to the final burnout point.

6.2.2.1.2. Circular Profile. Using the spherical coordinates notation of earlier sections,

$$\widehat{X}_i = (\widehat{r}_i,\ \widehat{\theta}_i,\ \widehat{\phi}_i) \sim (\widehat{x}_i,\ \widehat{y}_i,\ \widehat{z}_i)\ ,$$

where

$$\begin{aligned}
\widehat{x}_i &= \widehat{r}_i \sin\widehat{\phi}_i \cos\widehat{\theta}_i. \\
\widehat{y}_i &= \widehat{r}_i \sin\widehat{\phi}_i \sin\widehat{\theta}_i. \\
\widehat{z}_i &= \widehat{r}_i \cos\widehat{\phi}_i.
\end{aligned}$$

Consistent with our definition of constellations, the circular position profile is obtained as follows: First select an initial inclination-elevation-phasing triple $(\widehat{\gamma}_1,\ \widehat{r}_1,\ \widehat{\theta}_1^0)$ for cluster C_1. This equals the terminal (booster burnout) point of a preceeding linear phase. The remaining cluster positions are defined recursively as

$$(\widehat{\gamma}_i,\ \widehat{r}_i,\ \widehat{\theta}_i^0) = (\widehat{\gamma}_{i-1},\ \widehat{r}_{i-1},\ \widehat{\theta}_{i-1}^0) + (\Delta\widehat{\gamma}_i,\ \Delta\widehat{r}_i,\ \Delta\widehat{\phi}_i)\ , \tag{6.17}$$

where the perturbation terms are uniformly distributed in the intervals $[\gamma_a,\ \gamma_b]$, $[r_a,\ r_b]$ and $[\phi_a,\ \phi_b]$, respectively.

6.2.2.1.3. Random Profile. Cluster centroids for this shape are normally distributed about some normally distributed position $\widehat{X}_1$ of the first cluster C_1:

$$\begin{aligned}
\widehat{X}_i &= \widehat{X}_{i-1} + \widehat{W}_i\ , \\
\widehat{W}_i &= (\widehat{W}_{ix},\ \widehat{W}_{iy},\ \widehat{W}_{iz})\ , \\
\widehat{W}_{ix},\ \widehat{W}_{iy},\ \widehat{W}_{iz} &\approx N(\mu_i,\ \sigma_i)\ .
\end{aligned} \tag{6.18}$$

6.2.2.2. Cluster Centroid Velocity. Two types of velocity profile are considered to simulate the motion of target clusters, the *convergent linear dynamic* and the *convergent circular dynamic*. Target scheduling algorithms are not as sensitive to the detailed physical behavior of targets as tracking algorithms would be, and in constructing threat motions, THREATSIM is allowed to violate the laws of dynamics in order to obtain a synthetic threat that will stress the DDTS algorithm beyond any level obtainable with more "physical" simulators.

6.2.2.2.1. The Convergent Linear Dynamic. This dynamic does not contain any acceleration terms and is therefore an obvious simplification of what may happen during the boost phase, but it is satisfactory for scheduling purposes. If extended far enough into the future, it forces the centroids to converge along specified fixed unit vectors e_i. For the ith cluster,

$$\dot{\hat{X}}_i(t) = \eta_i e_i(1 - e^{-t/\tau}) + e^{-t/\tau}\dot{\hat{X}}_{io} \ , \ \ 0 \le t \le t_B \ , \tag{6.19}$$

where

$\eta_i \ \approx U[\eta_a, \ \eta_b], \ \eta_a < \eta_b.$

$\dot{\hat{X}}_{io}$ is the initial velocity of C_i (specified)

τ is the *convergence rate* factor.

6.2.2.2.2. The Convergent Circular Dynamic. To model the transition from the boost phase to the orbital phase, THREATSIM transitions (nonsmoothly) from the terminal state of a linear motion to a constant radius circular orbit. Let the duration of the boost phase be t_B. Then the position of C_i at time t_B is

$$\hat{X}_i(t_B) = \eta_i t_B e_i + \tau(\dot{\hat{X}}_{io} - \eta_i e_i)(1 - e^{-t_B/\tau}) \ ,$$

$$= (\hat{x}_i(t_B), \ \hat{y}_i(t_B), \ \hat{z}_i(t_B)) \ . \tag{6.20}$$

In spherical coordinates,

$$\hat{x}_i(t_B), = (\hat{r}_i(t_B), \ \hat{\theta}_i(t_B), \ \hat{\phi}_i(t_B)) \ ,$$

where

$$\hat{r}_i(t_B)^2 = \hat{x}_i^2(t_B) + \hat{y}_i^2(t_B) + \hat{z}_i^2(t_B) \ ,$$

$$\hat{\theta}_i(t_B) = tan^{-1}\left(\frac{\hat{y}_i(t_B)}{\hat{x}_i(t_B)}\right) \ ,$$

$$\hat{\phi}_i(t_B) = \cos^{-1}\left(\frac{\hat{z}_i(t_B)}{\hat{r}_i(t_B)}\right) \ .$$

The transition into circular orbit is done by clamping $\hat{r}_i(t)$ and the angular rates $\dot{\hat{\theta}}_i(t)$ and $\dot{\hat{\phi}}_i(t)$ at their terminal values (achieved at $t = t_B$) for all $t \ge t_B$.

The angular rates are computed using the relationships

$$\frac{d}{dt}\Big[\tan^{-1}\left(\frac{y(t)}{x(t)}\right)\Big] = \left(\frac{x^2(t)}{x^2(t) + y^2(t)}\right)\left(\frac{x(t)y'(t) - y(t)x'(t)}{x^2(t)}\right) \ ,$$

and

$$\frac{d}{dt}\Big[\cos^{-1}\left(\frac{u(t)}{v(t)}\right)\Big] = -\left(\frac{v(t)}{(v^2(t) - u^2(t))^{1/2}}\right)\left(\frac{(v(t)u' - u(t)v'(t)}{v^2(t)}\right) \ . \tag{6.21}$$

6.2.2.2.3. The Divergent Linear Dynamic. In an earlier section (see Eq. (6.16)) we described the linear initial position profile for cluster centroids. This profile has a mean direction vector $\overline{\widehat{v}}$. The divergent dynamic attempts to drive clusters away from $\overline{\widehat{v}}$ by forcing this motion towards a plane orthogonal to that vector, like Eq. (6.19), forces centroids towards the unit vectors e_i. This produces irregular threat profiles that tend to stress target scheduling algorithms since there is little structure to exploit in such expanding dynamical threats.

Starting with the reference vector $V \equiv \overline{\widehat{V}}$, consider any other vector V' such that $V' \neq kV$ for any real number k. Then $V' \times V$ is orthogonal to V, and $(V' \times V) \times V$ is orthogonal to both $V' \times V$ and V. Any vector in any plane orthogonal to V is thus a linear combination

$$W = \xi e_1 + \xi e_2 \,, \tag{6.22}$$

where

$$e_1 = \frac{V' \times V}{\| V' \times V \|} \,,$$

$$e_2 = \frac{(V' \times V) \times V}{\| (V' \times V) \times V \|} \,,$$

and ξ_1 and ξ_2 are two real numbers.

To generate divergent targets, we select any such vector V', and we sample ξ_1 and ξ_2 from the uniform distributions $U[\xi_{1a}, \xi_{1b}]$ and $U[\xi_{2a},\ \xi_{2b}]$.

6.2.3. Simulating Target Motion

Individual target positions and velocities are simulated as Gaussian perturbations on the centroid motion of their parent cluster. Each cluster C_i has M_i elements or targets,

$$C_i = \{T_{ij} : j = 1, \ldots, M_i\} \,,$$

whose positions and velocities are X_{ij} and $\dot{X}_{ij}$, respectively.

6.2.3.1. Target Positions

The positions of targets in a cluster C_i are modeled as Gaussian deviations whose means are uniformly distributed. For target T_{ij},

$$X_{ij} = \widehat{X}_i + \Delta X_{ij} \,, \tag{6.23}$$

where $\Delta X_{ij} = (\Delta X_{ijx},\ \Delta X_{ijy},\ \Delta X_{ijz})$, and $\Delta X_{iju} \approx N(\mu_{iju},\ \sigma_{iju})$, $u = x, y, z$.

6.2.3.2. Target Velocities. For velocities, a similar approach is used:

$$\dot{X}_{ij} = \hat{X}_i + \Delta\dot{X}_{ij} \,, \tag{6.24}$$

where $\Delta\dot{X}_{ij} = (\Delta\dot{X}_{ijx},\ \Delta\dot{X}_{ijy},\ \Delta\dot{X}_{ijz})$, and $\Delta\dot{X}_{iju} = N(\dot{\mu}_{iju},\ \dot{\sigma}_{iju})$.

6.2.4 Modeling Deadline and Release Times

We assume that all targets that belong to the same cluster have the same deadline and release time.

6.2.4.1 Deadline Simulation. Considering the enormous permutations of deadlines, that could be generated, some deadline structure is added by assuming that, *in the mean*, targets farther along their trajectory have less time-to-go than earlier targets. If d_i is the deadline of cluster C_i, $i-1,\ldots,k$, we employ the following simple additive scheme:

$$d_i = d_{i+1} + \Delta d_i,\ i = k-1,\ldots,1, \tag{6.25}$$

with $\Delta d_i \approx N(\mu_d,\ \sigma_d)$.

6.2.4.2 Release Time Simulation. Release times always precede deadlines (otherwise the cluster is rejected), and we assume that the opportunity window size is uniformly distributed. The following approach produces the desired result, with r_i the release time of cluster C_i.

$$r_i = d_i - W_i,\ i = 1,\ldots,k \,, \tag{6.26}$$

where $w_i \approx U[w_a,\ w_b]$.

6.3 Testing DDTS

Most of the accepted approaches to target scheduling are based on some version of the nearest neighbor (NN) algorithm,* so we used the NN algorithm as a benchmark against which the DDTS algorithm was compared. But we had to considerably enhance NN to include deadlines and release times, otherwise the comparison would be a bit unfair since NN could be made to miss almost all of its targets by selecting a sufficiently tight deadline structure.

Many threats were generated with THREATSIM to test the performance of the DDTS algorithm. In most cases, particularly for dense dynamic threats with severe deadlines, leakage risk for DDTS was considerably lower than for other algorithms. Using a mild threat consisting of 100 targets, we conducted 8 tests:

* Papadimitriou and Steiglitz (1982); Lawler et al. (1985).

1. Expected leakage risk versus defense reaction time.
2. Expected leakage risk versus engagement time.
3. Quantity of targets leaked versus initial beam position.
4. Expected leakage risk versus cluster centroid speed variance.
5. CPU running time versus threat size.
6. Risk versus probability of correct target identification.
7. Risk versus fast steering damping constant.
8. Risk versus fast steering motion limit.

6.3.1 Expected Leakage Risk versus Defense Reaction Time

Defense reaction time is defined as the time elapsed between first detection of the threat and the data handover from the battle manager to the DEW platform. Even for this relatively easy threat (Fig. 6.4), DDTS consistently dominates the enhanced NN algorithm, especially for small defense reaction times.

The principal reason for this dominance is that the NN method does not understand deadlines and release times, and it is unable to reschedule targets when such constraints are violated.

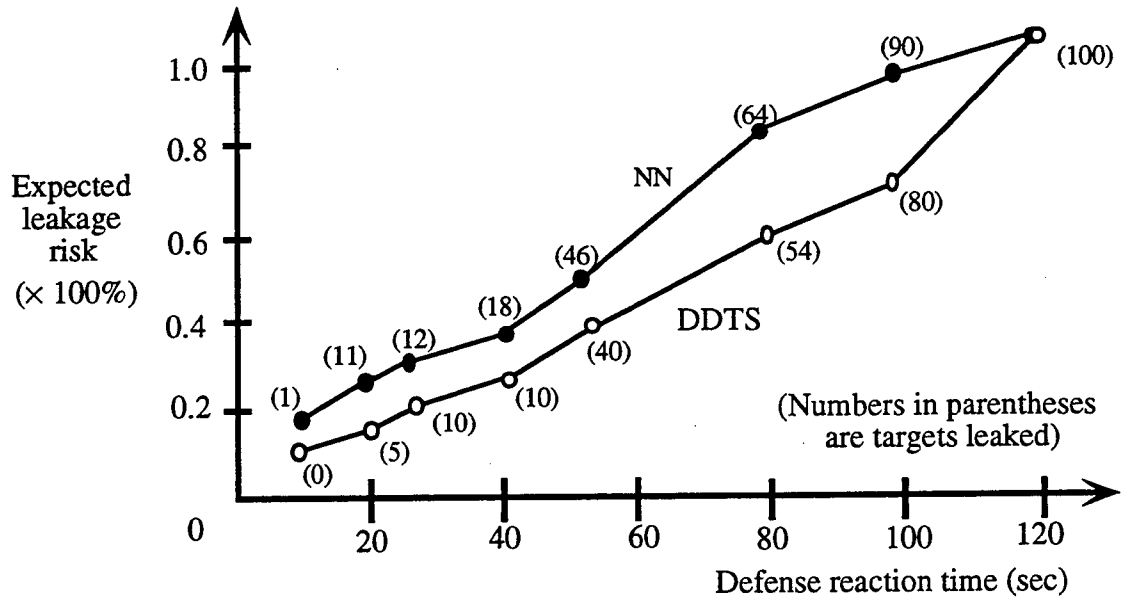

Figure 6.4. Even for a mild threat, DDTS reduces risk considerably.

As the reaction time approaches 120 seconds, the threat floats by the DEW platform unhindered, and all targets leak through.

6.3.2 Expected Leakage Risk versus Engagement Time

Engagement time is the time elapsed since data handover from the battle manager. Clearly, DDTS reduces the risk more rapidly than NN as the engagement evolves, gaining 10 seconds of threat processing time as 60 seconds have elapsed (Fig. 6.5). This gain is due to improved allocations of dwell times, shortened retarget times overall, and a better target rejection policy. Other target scheduling algorithms, whether based on NN principles or not, reject targets only because they are late and do not rearrange schedules to account for the relative importance of targets.

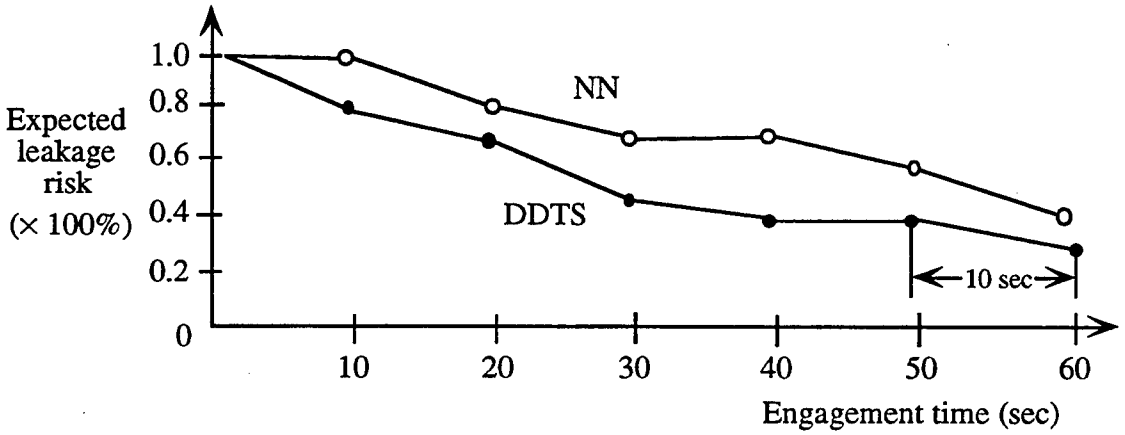

Figure 6.5. DDTS gets better answers faster.

6.3.3 Quantity of Targets Leaked versus Initial Beam Position

Referring to Fig. 6.6, observe that the NN algorithm is very sensitive to the initial aim direction of the DEW, essentially because it starts with the target nearest to that initial aimpoint, rather than slewing to the target nearest to its deadline. No targets leaked through with DDTS, whereas up to 16 leaked through with NN. This is understandable, since NN will typically sweep from the middle of the threat to one extremity, then return to the other extremity, thereby unnecessarily revisiting many points in space. Observe that the total dwell and retarget times were identical for both algorithms and showed no dependence upon the initial position of the beam.

Initial beam position	Leakers		Dwell time (sec)		Retarget time (sec)	
	DDTS	NN	DDTS	NN	DDTS	NN
Front	0	6	18.9	18.7	45.6	46.0
Middle	0	16	18.9	18.7	45.6	46.0
Rear	0	11	18.9	18.7	45.6	46.0

Figure 6.6. The algorithm is very insensitive to initial beam position.

6.3.4 Expected Leakage Risk versus Cluster Centroid Speed Variance

Each cluster was given a different centroid velocity in accordance with the laws of Eqs. (6.19) and (6.20). As the velocity or speed variance is increased, the threat is more difficult to handle since it is spreading through space, and predicting its future state becomes critical. Nearest neighbor approaches are very myopic in this regard, scheduling targets without due consideration of the future position at the time of their processing. The advantages of predicting future target positions are clear from Fig. 6.7, where the targets leaked by DDTS remain at zero as speed variance is increased, whereas those leaked by NN increase rapidly beyond 10 km/sec. Note the interesting leakage characteristics of DDTS below 5.0 km/sec due to the time-varying threat geometry. As the cluster speed variations are increased, the threat is first harder to handle, then easier. We were unable to explain exactly why this initial increase in risk for DDTS

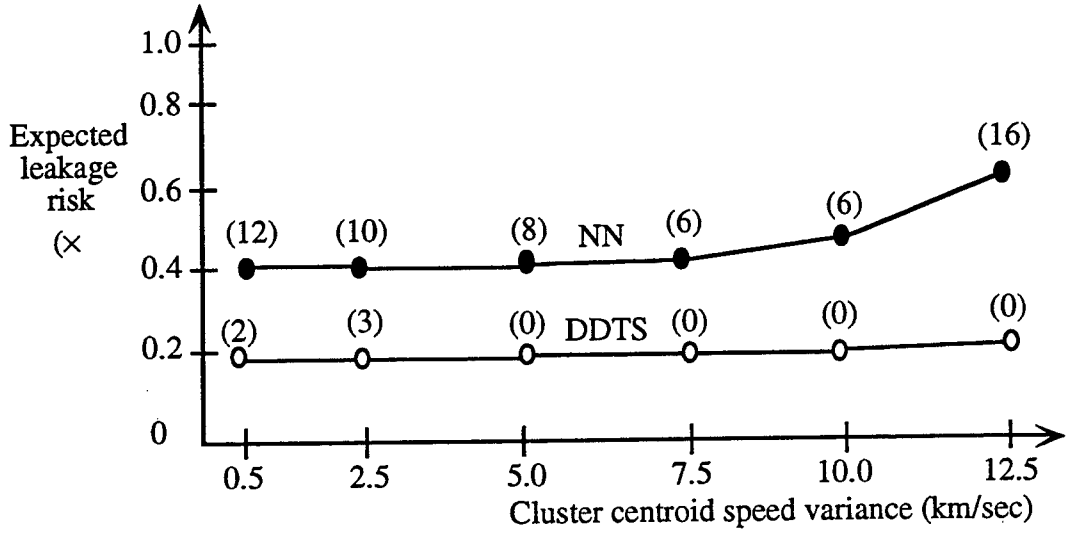

Figure 6.7. DDTS is much less sensitive to velocity effects.

disagrees—at least qualitatively—with the corresponding decrease for NN. We suspect that this minor deviation occurred because DDTS found a local minimum only at 2.5 km/sec.

6.3.5 CPU Running Time versus Threat Size

Both algorithms were run on a Silicon Graphis IRIS machine. Observe that, for anything but trivial threats, targets will be rescheduled several times, so that there is no point scheduling targets too far into the future. To avoid redundant work, we never schedule more than k targets at a time, where k is called the "look-ahead" number. The worst-case complexity of DDTS is $O(n^3)$, where n is the quantity of targets scheduled, but several experiments with k indicate that the target completion time actually varies as $k^{2.5}$, a slight improvement over the worst case.

Again, for most threats, DDTS was faster than NN, so a threat was chosen which challenges all the subroutines of DDTS, the 2-opt subroutine especially, and the graph shown in Fig. 6.8 reports the resulting completion time.

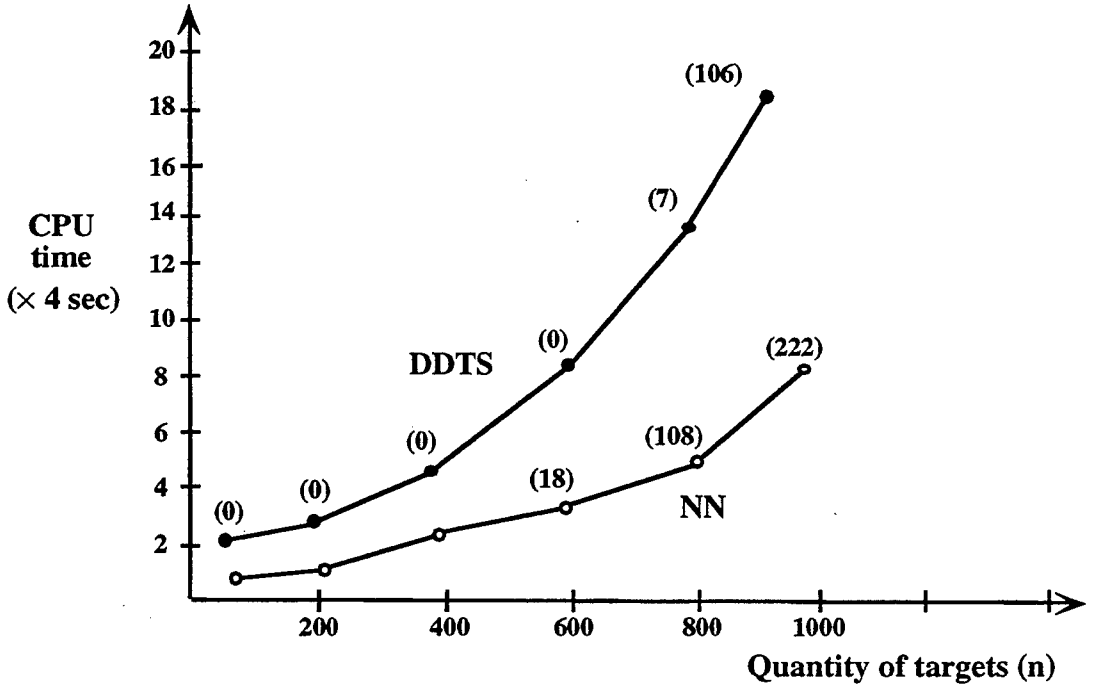

Figure 6.8. In spite of considerable physics and optimization work, the DDTS algorithm is efficient in time and space.

The graph was obtained for $k = 100$, a rather large number considering the fluidity of the threat. Note that the implementation was on a sequential machine (IRIS), but is intended for parallel implementation on actual platforms. Also, programming was not

optimized; however, a speed-up of a factor of about 2 to 4 can be obtained by passing the current version of DDTS through a code optimizer. The Brent optimization subroutine performed admirably, requiring an average of only 4 iterations to reach optimality. The space requirements for DDTS are minimal because the dynamic memory allocation features of the C language are exploited and only a few target sequences are kept in storage at any given time.

6.3.6 Risk versus Probability of Correct Target Identification

The costs associated with target misidentification are severe. Target vulnerability or hardness is misjudged, resulting in erroneous dwell times and probable reductions in p_K. Aimpoint location errors are usually made as well, and these further reduce p_K. For the tests reported here, we made the conservative assumption that any misidentified targets leak through; Figure 6.9 shows that p_{ID} has a significant effect on platform performance, so that reliable target identification algorithms must be used to assure acceptable platform performance.

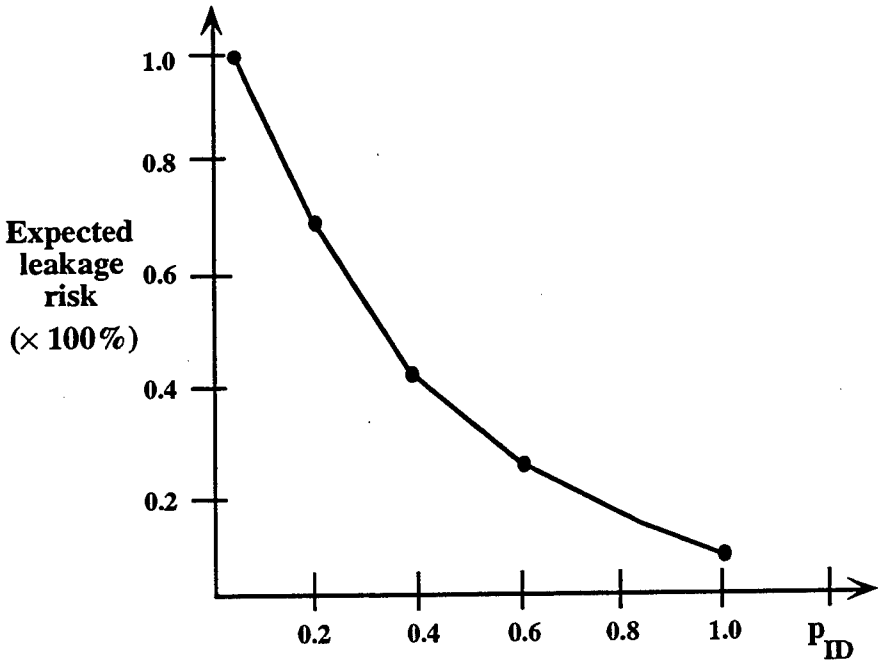

Figure 6.9. Platform performance is very sensitive to correct target identification.

6.3.7 Risk versus Fast Steering Damping Constant

In DDTS, platform hardware dynamics are modeled as second-order linear systems. When control or steering components are underdamped, their settling time is usually high, and this can severely increase retarget times, resulting in increased leakage risk.

Fortunately, as shown in Fig. 6.10, no severe harm is done, at least as far as the fast steering system is concerned, until damping constants fall below 0.04.

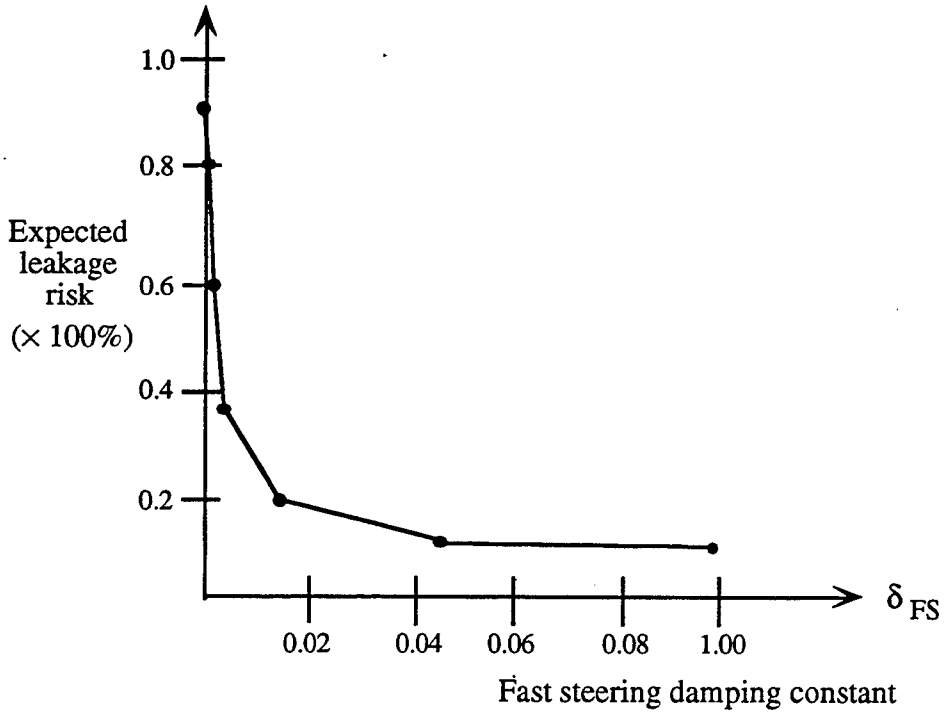

Figure 6.10. System performance is not sensitive to fast steering damping.

6.3.8 Risk versus Fast Steering Motion Limit

Motion limits of beam control and steering components affect performance only when the angular distance between targets is large, which typically occurs only when threats are rather sparse. For the threats used in these experiments, the fast steering motion limits had no significant impact on performance, as the rather comforting but uninteresting graph of Fig. 6.11 indicates. The DDTS algorithm obviously sheduled targets in such a way that few if any retarget angles exceeded a few milliradians. The inclusion of hardware dynamics and constraints in the DDTS algorithm was essential to minimize retarget times.

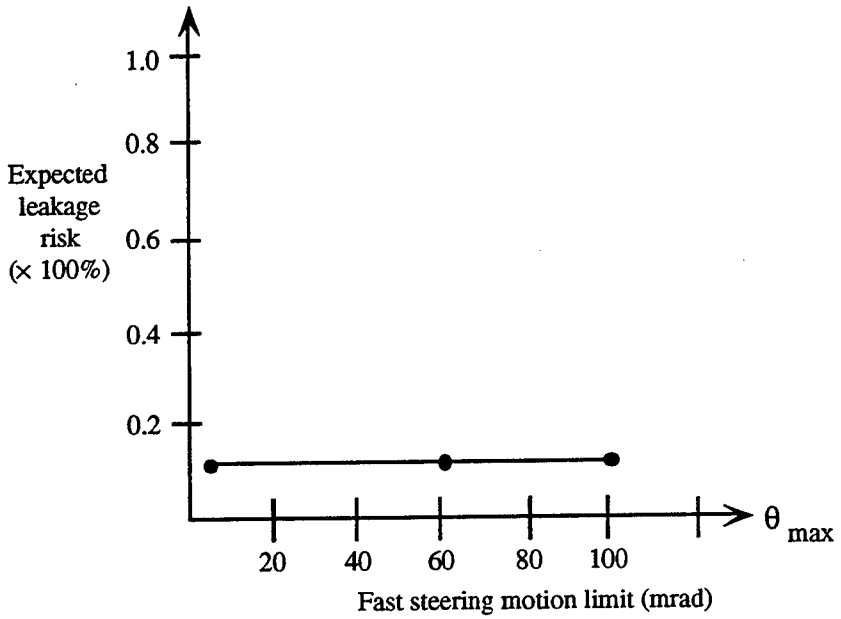

Figure 6.11. Motion limits had no effect on performance for the threat considered.

7. Conclusions and Future Work

During this three-year program, we have developed, tested, and delivered algorithms for managing HEL and NPB platforms operating in earth orbit during the boost and midcourse phases, with specific emphasis on the optimal allocation of weapon resources. The resource allocation problem was expressed as a scheduling problem where the prosecution of a potentially large collection of targets must be planned so as to minimize the total cost of target leakage, the *Expected Leakage Risk*. This planning activity determines in what order targets must be processed, how much processing time should be allocated to each target, and which targets should be ignored when there is insufficient time to process all targets by their deadlines.

From our detailed analyses and experiments, we conclude that the target scheduling problem indeed turned out to be much more complicated than was expected at the outset. Previous scheduling approaches, for instance, have ignored or neglected several operational conditions and constraints that strongly influence the performance of DEW platforms in both the boost phase kill and in the midcourse discrimination phase. The elimination of such shortcomings forced us to conclude that previous studies had also been too abstract and could have been more "physics-based". But the inclusion of better physics models also painfully increased the computational complexity of conventional scheduling procedures, so that some technology improvements were needed to obtain an algorithm whose time and space behavior would be acceptable for this real-time application. Based on a balanced mixture of formal and heuristic methods, the Deadline Driven Target Scheduling (DDTS) algorithm satisfies the required operational constraints and provides excellent performance and computational efficiency.

Fearing that too much modeling detail had been included in DDTS (over parametrization), we ran exhaustive sensitivity tests using THREATSIM, a threat driver specifically designed to exercise DDTS. We confirmed that earlier algorithms had ignored physical phenomena whose inclusion in DDTS not only improved platform performance but also influenced the internal structure of the optimization subroutines. But we could not confirm that our chosen level of modeling detail is optimal in any sense. We are now in fact certain that it is not because we have discovered during this writing that platform functions contiguous to target scheduling may influence the schedules. Recent unpublished research in DEW aimpoint selection (Probst 1992), for instance, indicates that satisfactory aimpoint selection is not possible for certain threat configurations. Targets should obviously not be scheduled for processing during configurations where no suitable aimpoint can be found. Consequently, one important issue to be addressed during further research is to integrate target scheduling algorithms with other platform functions such as Aimpoint Selection and Aimpoint Maintenance.

Other than the rather obvious fact that deadlines constitute the most severe constraint when threats are at least moderately dense, no general conclusions about threat complexity were sought during this research. The DDTS algorithm simply does the best that is reasonably achievable with all the available information, and within the stated operational constraints.

Acknowledgments

This research was sponsored by the Strategic Defense Initiative Organization (SDIO) of the Department of Defense (DoD); it was directed by Mr. William A. Fontana, Project Engineer in the OCDE Section of the Rome Laboratory (RL) at Griffiss Air Force Base. This sponsorship is sincerely appreciated.

The author would like to thank Mr. Kevin Probst for initiating and supporting this program during his tenure at SDIO, and Mr. Fontana for his continued guidance throughout this three-year research period. His assistance in providing important technical connections with other technology development programs was helpful in our attaining a sufficiently broad point of view for the target scheduling problem.

Although too numerous to mention individually, the author would like to thank both contractor and government personnel working on various fire control, ATP, and simulation testbed development programs for their comments and general recommendations on this program. I would like, however, to thank a few specific individuals with whom technical interactions were especially fruitful. To Mr. Gary Gurski of the General Research Corporation (GRC), many thanks for sharing with me the extensive fire control work done at GRC: that work provided a flying start for my research. The battle management recommendations provided by Dr. Sheldon Cantor of the Aerospace Corporation are also appreciated. Sincere thanks to Dr. Thomas Wehner of the Los Alamos National Laboratory for providing the physical insight needed to develop the conceptual NPB models used in this report.

I am grateful to Dr. R. Glaser (LLNL) for the careful review of the mathematical and computational concepts on which the DDTS algorithm is based, and for his thorough derivation of some of the functional relationships. I am particularly endebted to him for his flawless C-code version of DDTS, and to Mr. Richard Gassner (RL) for his β-testing of the algorithm.

Appendices

Appendix A.

Derivation of Derivatives

We derive the angular derivatives $\dot{\theta}^E(t_o)$ and $\dot{\phi}^E(t_o)$ used in Eq. (2.28) of Chapter 2. Referring to Fig. A.1, and omitting E-subscripts,

$$| r | = \sqrt{(x)^2 + (y)^2 + (z)^2} \, ,$$

$$\theta = tan^{-1}(y/x) \, ,$$

$$\phi = cos^{-1}\left(\frac{z}{\sqrt{x^2 + y^2 + z^2}}\right) .$$

Define $w \triangleq x/y$.

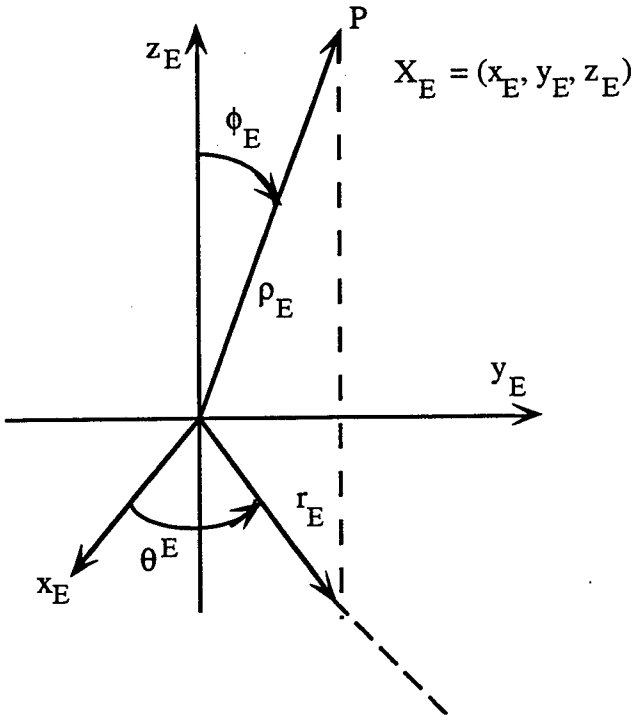

Figure A.1. Coordinate transformation diagram.

A.1 Computing $| \dot{r} |$

$$\frac{d \mid r \mid}{dt} = \frac{2(x\dot{x} + y\dot{y} + z\dot{z})}{\sqrt{x^2 + y^2 + z^2}} \,. \tag{A.1}$$

A.2 Computing $| \dot{\theta} |$

$$\dot{\theta} = \frac{d(tan^{-1}w)}{dt} = \left(\frac{1}{1 - w^2}\right)\frac{dv}{dt} \,,$$

$$= \frac{x\dot{y} - y\dot{x}}{x^2(1 + w^2)} = \frac{x\dot{y} - y\dot{x}}{x^2(1 + (\frac{y}{x})w^2)} \,,$$

Hence:

$$\dot{\phi} = \frac{x\dot{y} - y\dot{x}}{x^2 + y^2} \,. \tag{A.2}$$

A.3 Computing $| \dot{\phi} |$

$$\dot{\phi} = \frac{d(cos^{-1}v)}{dt} = -\left(\frac{1}{\sqrt{1 - v^2}}\right)\frac{dv}{dt} \,,$$

where

$$v = \frac{z}{\sqrt{x^2 + y^2 + z^2}} \,,$$

$$\frac{dv}{dt} = \left(\frac{\sqrt{x^2 + y^2 + z^2}\right)\dot{z} - z\left[\frac{\dot{x}x + \dot{y}y + \dot{z}z}{\sqrt{x^2 + y^2 + z^2}}\right]}{(x^2 + y^2 + z^2)} \,,$$

$$= \frac{\dot{z} - z\left[\frac{\dot{x}x + \dot{y}y + \dot{z}z}{x^2 + y^2 + z^2}\right]}{(x^2 + y^2 + z^2)^{1/2}} \,,$$

$$= \frac{\dot{z}}{(x^2 + y^2 + z^2)^{1/2}} - \frac{z(x\dot{x} + y\dot{y} + z\dot{z})}{(x^2 + y^2 + z^2)^{3/2}} \,.$$

But

$$\frac{1}{\sqrt{1-v^2}} = \frac{(x^2+y^2+z^2)^{1/2}}{(x^2+y^2)^{1/2}}.$$

Combining,

$$\dot{\phi} = \frac{z(x\dot{x}+y\dot{y}+z\dot{z})}{(x^2+y^2+z^2)\cdot(x^2+y^2)^{1/2}} - \frac{\dot{z}}{(x^2+y^2)^{1/2}},$$

$$= \frac{1}{\sqrt{x^2+y^2}}\left[\frac{z(x\dot{x}+y\dot{y}+z\dot{z})}{(x^2+y^2+z^2)} - \dot{z}\right]. \qquad (A.3)$$

Appendix B.
Derivation of the Normal Approximation for P(Z > z)

Recall from the text in Section 3.3.1.2 (see Eq. (3.25)):

$$p(Z > z) = \exp\Big(-\frac{\mu^2}{2\sigma_o^2}\Big) \int_z^\infty \frac{v}{\sigma_o^2} \exp\Big(\frac{-v^2}{2\sigma_o^2}\Big) \sum_{n=0}^{\infty} \frac{1}{(n!)^2} \Big(\frac{\mu v}{2\sigma_o^2}\Big)^{2n} dv \,. \qquad (B.1)$$

Changing variables, let $w = v^2/2\sigma_o^2$, then $dw = 2vdv/2\sigma_o^2 = v/2\sigma_o^2 dv$. Thus,

$$p(Z > z) = \exp\Big(-\frac{\mu^2}{2\sigma_o^2}\Big) \int_{z^2/2\sigma_2^0}^\infty e^{-w} \sum_{n=0}^{\infty} \frac{1}{(n!)^2} \Big(\frac{\mu^2}{2\sigma_0^2}\Big)^n w^n \, dw \,,$$

$$= \exp\Big[-\frac{\mu^2}{2\sigma_0^2}\Big] \sum_{k=0}^{\infty} \frac{1}{(n!)^2} \Big(\frac{\mu^2}{2\sigma_0^2}\Big)^n \int_{z^2/2\sigma_2^0}^\infty e^{-w} \, w^n dw.$$

But

$$\int_S^\infty e^{-w} w^n dw = n! \int_S^\infty \frac{1}{n!} w^{(n+1)-1} e^{-w}$$

$$dw = n! \sum_{l=0}^{n} \frac{e^{-s} s^l}{l!} \,.$$

Substituting, and letting $\alpha = \mu^2/2\sigma_0^2$,

$$p(Z > z) = \exp(-(\alpha + s)) \sum_{n=0}^{\infty} \frac{\alpha^n}{n!} \sum_{l=0}^{n} \frac{s^l}{l!} \,,$$

$$= \Big(\sum_{n=0}^{\infty} \frac{\alpha^n e^{-a}}{n!}\Big)\Big(\sum_{l=0}^{n} \frac{s^l e^{-s}}{l!}\Big) \,,$$

$$= \text{prob}(X_s \le X_\alpha) = \text{prob}\,(X_s - X_v \le 0), \qquad (B.2)$$

where X_s and X_α are independent Poisson variables with parameter s and α, respectively.

If we approximate each Poisson variable by a normal variable with a corrected mean and same variance, then

$$\text{prob}(X_s - X_v \le 0) = \text{prob}\,(N_s - N_v < 0) = \text{prob}(N \le 0),$$

where

$$N \sim N\Big(s - v - 1/2, \sqrt{s+v}\Big). \tag{B.3}$$

Hence,

$$p(Z > z) = \Phi\Big(\frac{1/2 - (s-v)}{\sqrt{s+v}}\Big) .$$

Resubstituting for $s = z^2/2\sigma_o^2$, $v = \mu^2/2\sigma_o^2$ and $\sigma_o = R\sigma$,

$$p(Z > z) = \Phi\Big(\frac{(R\sigma)^2 - d^2 + \mu^2}{R\sigma\sqrt{2(d^2+\mu^2}}\Big) . \tag{B.4}$$

Appendix C.
Deriving the Dwell Time t^{D*} from the Kill Probability p_K^*

In Section 3.3.1.4 (see Eq. (3.31)) a derivation of t^{D^*} from p_K^* was presented, but a few mathematical details were intentionally deferred. These details are presented in this appendix. Even though all the required symbols and definitions were introduced in Section 2.3.3.1, a slightly different set of terms is used here to simplify exposition and to provide a self-contained discussion.

Objective: Obtain expression for dwell time in terms of kill probability.

Notation

p_K	kill probability
t_D	dwell time
μ	mean hardness of target
σ	standard deviation of target hardness
L	$\mu - 1.5\sigma$
U	$\mu + 1.5\sigma$
p_0	probability (miss target) (beam circle $\cap$ target circle $= \phi$)
p_1	probability of maximum target coverage by beam
	(beam circle $\subset$ target circle or target circle $\subset$ beam circle)
r_{DEW}	radius of beam circle $= 4R\lambda/\sqrt{\pi}D$ (R, λ, D defined below)
r_{AIM}	radius of target circle
r_1	max (r_{DEW}, r_{AIM})
r_2	min (r_{DEW}, r_{AIM})
R	distance from DEW to target
$A°$	πr_2^2 maximum possible area of target coverage by beam
a	angle of incidence of beam on target ($\alpha = 0$ is optimal hit)

P	DEW power
λ	wavelength
D	optics telescope size
K_T	incidence angle damping constant
Q	$\frac{\pi P D^2 \cos a}{4\lambda^2 R^2 \exp[R^2/K_T \cos\alpha]}$
θ	Qt_d/a°
q_0	$1 - p_0$, probability target is hit (at least partially) by beam

Define two random variables:

A = area of target coverage by beam (area of beam circle $\cap$ target circle)

H = hardness of target

Kill occurs if, and only if, $\theta A > H$.

Therefore,

$$\begin{aligned} p_K &= p(\theta A \geq H) \,, \\ &= \int P(\theta A \geq H) \,|_{H=h}) f_H(h) dh \;\; (f_H \text{ is the pdf of H}) \,, \\ &= \int P(\theta A \geq h) f_H(h) dh \,, \quad \text{since } A \text{ and } H \text{ are independent} \,, \\ &= \int P(A \geq \frac{h}{\theta}) f_H(h)\, dh \,. \end{aligned} \tag{C.1}$$

Since $\theta = Qt_d/A^\circ$, evaluation of the above integral yields a relation between p_K and t_d. Clearly p_K increases with θ (and therefore with t_d). Our goal is to determine, for fixed p_K, the corresponding θ (and therefore t_d).

For computational efficiency, we use approximations to the distributions of H and A.

Approximation for H: Assume H is uniformly distributed over a 3 standard deviation range, $[L, U] = [\mu - 1.5s, \mu + 1.5s]$. Therefore,

$$f_H(h) = \begin{cases} \dfrac{1}{3\sigma} & if L \leq h \leq A \\ 0 & \text{otherwise} \end{cases} .$$

Approximation for A:

$$A = \begin{cases} 0 \text{ with probability } p_0 \\ A^\circ \text{ with probability} p_1 \\ \text{uniformly distributed on the range}(0, A^\circ) \end{cases}$$

That is, the CDF of A is (Fig. C.1)

$$P(A \geq s) = \begin{cases} 0 \text{ if } S < 0 \\ p_0 + \frac{S}{A^\circ}\ (q_0 - p_1) \quad if\ 0 \leq\ s\ < A^\circ\ . \\ 1 \text{ if } s \geq A^\circ \end{cases}$$

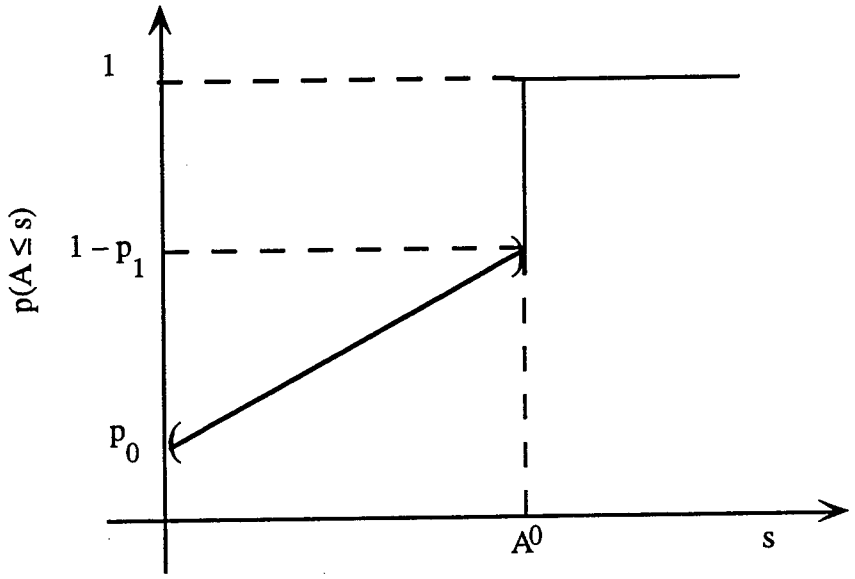

Figure C.1. Plot of the CDF of A ($q_0 = 1 - p_0$).

Therefore Eq. (C.1) simplifies to Eq. (C.2):

$$p_K = \frac{1}{3\sigma} \int_L^u P(A \geq \frac{h}{\theta}) dh\ . \qquad (C.2)$$

where (Fig. C.2)

$$P(A \leq \frac{h}{\theta}) = \begin{cases} 1 \text{ if } h \leq 0\ . \\ q_0 - \frac{h}{A^\circ \theta}(q_0 - p_1) \quad if\ 0\ <\ h\ \leq A^\circ\theta\ . \\ 0 \text{ if } h > A^\circ\theta \end{cases}$$

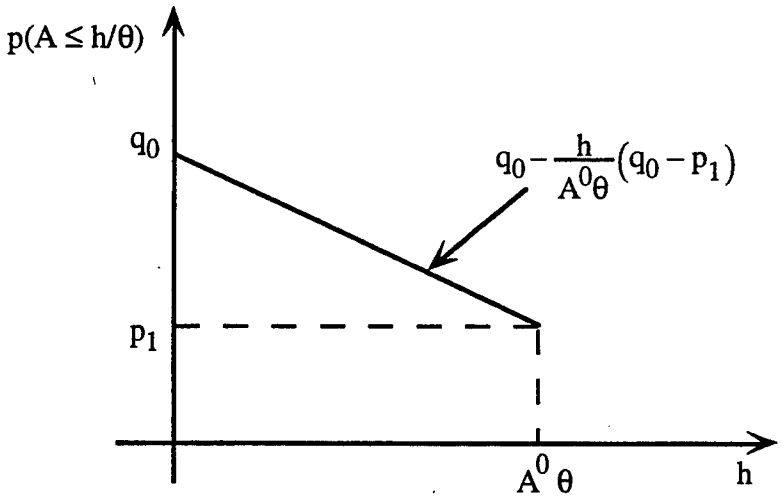

Figure C.2. CDF of A.

The case $h \leq 0$ does not arise because $L > 0$ by assumption.

Expression (C.2) determines p_K as a function of θ (or t_d). For $\theta \leq L/A^\circ$, p_K is identically 0 since then $h/\theta \geq A^\circ$. We may dismiss this situation since by assumption $p_K > 0$.

For $\theta > L/A^\circ$, p_K increases monotonically with θ, approaching q_0 as $q \to \infty$ (infinite dwell time). We fix p_K in $(0, q_0)$ and solve for θ in $(L/A^\circ, \infty)$.

Evaluation of Eq. (C.2) hinges on the locations of L and U.

Case 1: $L < A^\circ\theta \leq U$.

Here Eq. (C.2) reduces to

$$p_K = \frac{1}{3\sigma}\int_L^{A^\circ\theta}[q_0 - \frac{h}{A^\circ\theta}(q_0 - p_1)]dh\,. \qquad (C.3)$$

By monotonicity of p_K, the condition $L < A^\circ\theta \leq U$ is equivalent to $0 < p_K \leq p_K^0$, where p_K^0 is the value of p_K for $\theta = u/A^\circ$:

$$p_K^0 = \frac{1}{3\sigma}\int_L^U \left[q_0 - \frac{h}{U}(q_0 - p_1)\right]dh = \frac{1}{3\sigma}\left[q_0(U-L) - \frac{q_0 - p_1}{2U}(U^2 - L^2)\right],$$

$$= \frac{1}{3\sigma}\left[q_0(3\sigma) - \frac{q_0 - p_1}{2U}(2U)(3\sigma)\right], \quad \text{since } U - L = 3\sigma, U + L = 2\mu,$$

$$= q_0 - (q_0 - p_1)\frac{\mu}{U}.$$

To solve Eq. (C.3) for θ, we let $\beta = A^{\circ}\theta$ and solve for β in (L, U).

$$\text{Therefore, } p_K \frac{1}{3\sigma}\left[(q_0(\beta - L) - \frac{q_0 - p_1}{2\beta}(\beta^2 - L^2)\right], \qquad (C.4)$$

which yields the quadratic,

$$(q_0 + p_1)\beta^2 - 2(z_0 L + 3\sigma p_K)\beta + (q_0 - p_1)L^2 = 0\,. \qquad (C.5)$$

There are two roots to Eq. (C.5):

$$\beta = \frac{q_0\, L + 3\sigma p_K \pm \sqrt{(q_0\, L + 3\sigma p_K)^2 - (q_0 + p_1)(q_0 - p_1)L^2}}{q_0 + p_1}.$$

We now show that the "+" root is the desired solution in (L, U) to Eq. (C.4). View the right-hand side of Eq. (C.4) as a function of β, denoted $r(\beta)$. Then

$$\frac{d}{d\beta} r(\beta) = \frac{1}{3\sigma}\left[\frac{q_0 + p1}{2} - \frac{q_0 + p_1}{2} L^2 \frac{1}{\beta^2}\right], \qquad (C.6)$$

and

$$\frac{d^2}{d\beta}\, r(\beta) = \frac{1}{3\sigma}(q_0 - p_1)\, \frac{L^2}{\beta^3}\,. \qquad (C.7)$$

We are interested only in $\beta > 0$. Here $r(\beta)$ is concave up (from Eq. (C.7)) and is decreasing for $\beta < \alpha L$ and increasing for $\beta > \alpha L$ (from Eq. (C.6)), where

$$\alpha \equiv \sqrt{\frac{q_0 - p_1}{q_0 + p_1}} < 1.$$

Of the two solutions r_1 and r_2 to $r(\beta) = p_K (\beta > 0)$ (Fig. C.3), r_1 is not in (L, U) because $r_1 < \alpha L < L$. Since a solution to $r(\beta) = p_K$ in (L, U) exists, it must be r_2. Furthermore, since r_1 and r_2 are the two roots to Eq. (C-5), r_2 must be the larger one—i.e., the "+" root (Fig. C.3).

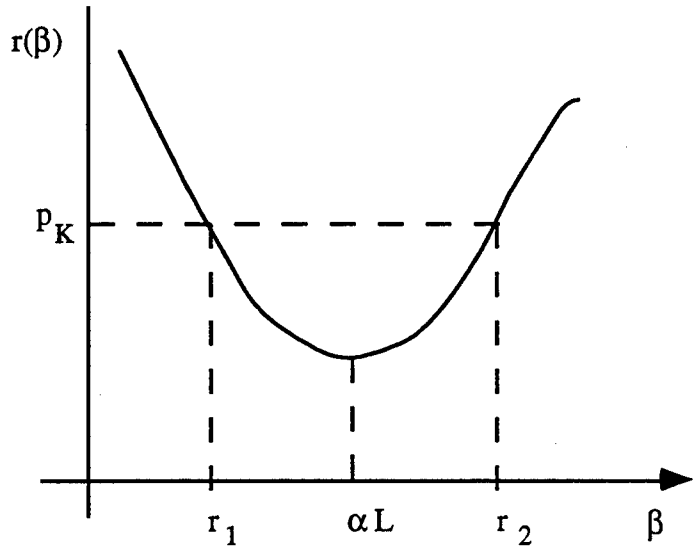

Figure C.3. Plot of $p_K(\beta > 0)$.

Recall that $\beta = A^0 q$. The Case 1 solution is therefore

$$\theta = \frac{1}{A^\circ}\left\{\frac{q_0 L + 3\sigma p_K + \sqrt{(q_0 L + 3\sigma p_K)^2 - (q_0 + p_1)(q_0 - p_1)L^2}}{q_0 + p_1}\right\}.$$

Case 2: $U < A^0\theta$, or equivalently $q_0 > p_K > p_K^0 = q_0 - (q_0 - p_1)\mu/U$. Hence Eq. (C.2) reduces to

$$p_K = \frac{1}{3\sigma}\int_L^U [q_0 - \frac{h}{A^\circ\theta}(q_0 - p_1)]dh. \qquad (C.8)$$

To solve Eq. (C.8) for θ, let $\beta = A^0\theta$ and solve for β in (U, ∞).

$$\begin{aligned} p_K &= \frac{1}{3\sigma}\left[(q_0(U - L) - \frac{q_0 - p_1}{2\beta}(U^2 - L^2)\right], \\ &= \frac{1}{3\sigma}\left[(3\sigma q_0 - \frac{q_0 - p_1}{2\beta}(2\mu)(3\sigma)\right], \\ &= q_0 - \frac{\mu(q_0 - p_1)}{\beta}. \end{aligned}$$

Therefore $\beta = \mu(q_0 - p_1)/q_0 - p_K)$, and the Case 2 solution is

$$\theta = \frac{\mu(q_0 - p_1)}{A^\circ(q_0 - p_K)}.$$

Recall that $t_d = A \circ \theta / Q$. We summarize our results.

The dwell time t_d corresponding to kill probability p_K is given by

$$t_d = \begin{cases} \dfrac{1}{Q}\left\{\dfrac{q_0 L + 3\sigma p_K + \sqrt{(q_0 L + 3\sigma p_K)^2 - (q_0 - p_1)(q_0 - p_1)L^2}}{q_0 + p_1}\right\} & , \text{ if } 0 < p_K \leq p_K^0 , \\ \dfrac{1}{Q}\left\{\dfrac{\mu(q_0 - p_1)}{q_0 + p_K}\right\} & , \text{ if } p_K^0 \leq p_K < q_0 , \end{cases} \tag{C.9}$$

where $p_K^0 = q_0 - (q_0 - p_K)\mu / U$.

Appendix D.
Heaps

We provide a more detailed but informal discussion of heaps and the HEAPSORT routine (read Press et al. (1988) for an expanded treatment).

Heapsort is a popular sorting routine. It can be recommended wholeheartedly for a variety of sorting applications. It is a true "in-place" sort, requiring no auxiliary storage. It is an $Nlog2N$ process, not only on average, but also for the worst-case order of input data. In fact, its worst case is only 20 percent or so worse than its average running time.

It is beyond our scope to give a complete exposition on the theory of Heapsort. We will mention the general principles. If you want to understand the detail, we refer you to Knuth (1973), or suggest you analyze the program yourself.

A set of N numbers a_i, $i = 1, \ldots, N$, is said to form a "heap" if it satisfies the relation

$$a_j/2 \geq a_j \quad \text{for} \quad 1 \leq j/2 < jN \,. \tag{D.1}$$

Here the division in $j/2$ means "integer divide": i.e., it is an exact integer, or else it is rounded down to the closest integer. Definition (D1) will make sense if you think of the numbers a_i, as being arranged in a binary tree, with the top ("boss") node being a1, the two "underling" nodes being a_2 and a_3, *their* four underling nodes being a_4 through a_7, etc. (See Fig. D.1). In this form, a heap has every "supervisor" greater than or equal to its two "supervisees", down through the levels of the hierachy.

If you have managed to rearrange your array into an order that forms a heap, then sorting it is very easy: You pull off the "top of the heap", which will be the largest element yet unsorted. Then you "promote" to the top of the heap its largest underling. Then you promote its largest underling, and so on. The process is similar to what happens (or is supposed to happen) in a large corporation when the chairman of the board retires. You then repeat the whole process by retiring the new chairman of the board. Evidently the whole thing is an $Nlog_2N$ process, since each retiring chairman leads to log_2N promotions of underlings.

To arrange the array into a heap in the first place, a "sift-up" process is similar to corporate promotion. Imagine that the corporation starts our with N/2 employees on

the production line, but with no supervisors. Now a supervisor is hired to supervise two workers. If he is less capable that both his workers, one of them is promoted in his place, and he joins the production line. After supervisors are hired, then supervisors of supervisors are hired, and so on up the corporate ladder. Each employee is brought in at the top of the tree, but then immediately sifted down, with more capable workers promoted until their proper corporate level has been reached.

In the Heapsort implementation, the same "sift-down" code can be used for the initial creation of the heap and for the subsequent retirement-and-promotion phase. One execution of the Heapsort function represents the entire life-cyle of a giant corporation: N/2 workers are hired; N/2 potential supervisors are hired; there is a sifting up in the ranks, a sort of Parkinson's Law; finally, in due course, each of the original employees gets promoted to chairman of the board.

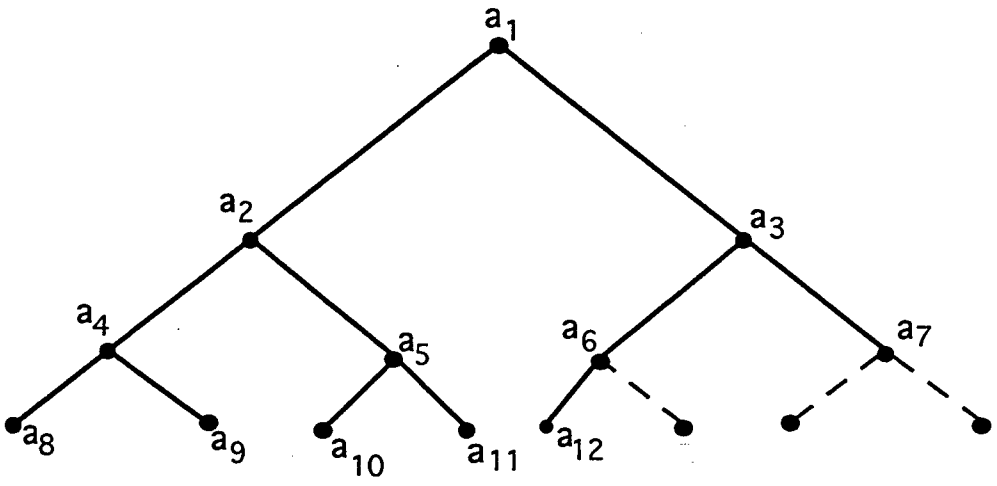

Figure D.1. Ordering implied by a "heap", here of 12 elements. Elements connected by an upward path are sorted with respect to one another, but there is not necessarily any ordering among elements related only "laterally".

Appendix E.

Computing p(X < Y)

Consider the cumulative distribution functions below in Figs. E.1 and E.2.

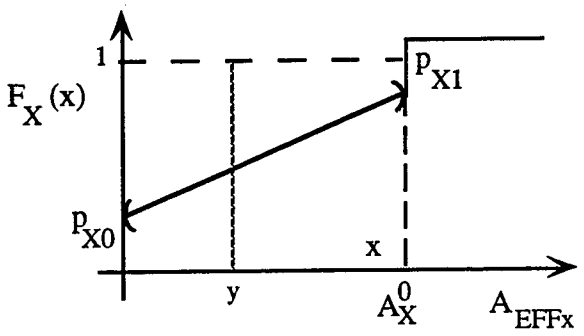

Figure E.1. CDF of A_{EFFX}.

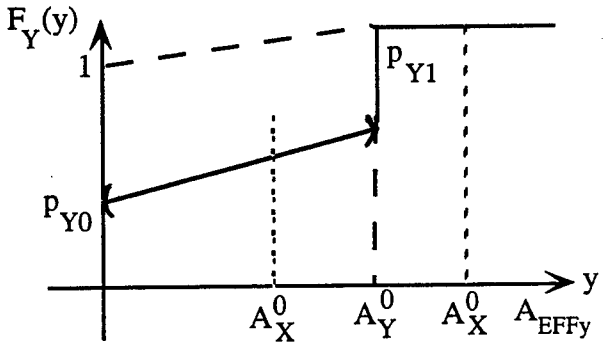

Figure E.2. CDF of A_{EFFY}.

The corresponding densities are

$$p_X(x) = p_{X0}\delta(x) + p_{X1}\delta(x - A_X^0) + \left(\frac{1 - p_{X0} - p_{X1}}{A_X^0}\right), 0 \le x \le A_X^0 , \qquad (E.1)$$

$$p_Y(y) = p_{Y0}\delta(y) + p_{Y1}\delta(y - A_Y^0) + \left(\frac{1 - p_{Y0} - p_{Y1}}{A_Y^0}\right), 0 \le y \le A_Y^0 , \qquad (E.2)$$

and they equal zero outside these regions.

Define $Z = X - Y$.

The problem is to find $p(Z \leq 0)$ (Fig. E.3).

$$p(Z \leq 0) = \int_{-\infty}^{\infty} \int_{-\infty}^{y} P_{XY}(x,y)dxdy = \int_{-\infty}^{\infty} f(y)p_Y(y)dy \; . \qquad (E.3)$$

$$f(y) = \int_{-\infty}^{y} p_X(x)dx = \begin{cases} 0 & , \; y < 0 \\ p_{X0} + \left(\dfrac{1 - p_{X0} - p_{X1}}{A_X^0}\right) y & , \; 0 \leq y < A_X^0 \\ 1 & , \; A_X^0 \leq y \end{cases} \; . \qquad (E.4)$$

$$\int_{-\infty}^{\infty} f(y)p_Y(y)dy = \int_{0}^{A_X^0 - \epsilon} \left(p_{X0} + \frac{(1 - p_{X0} - p_{X1})0}{A_X^0}\, y\right) p_Y(y)dy + \int_{A_X^0 - \epsilon}^{\infty} p_Y(y)dy \; . \qquad (E.5)$$

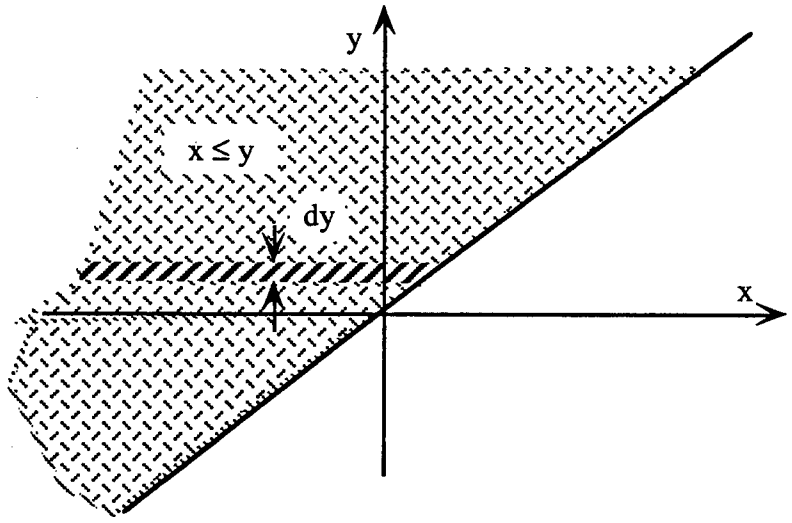

Figure E.3. Calculating $p(X \leq Y)$.

Case 1: $A_X^0 \leq A_Y^0$

$$p(Z \leq 0) = \int_{0}^{A_X^0 - \epsilon} \left(p_{X0}p_{Y0} + p_{X0}\left(\frac{1 - p_{Y0} - p_{Y1}}{A_Y^0}\right) + , \right.$$
$$\left. \left(\frac{(1 - p_{X0} - p_{Y0})}{A_X^0}\, \frac{1 - p_{Y0} - p_{Y1}}{A_Y^0}\right) y \right) dy \; ,$$

$$+\Big(\frac{1-p_{Y0}-p_{Y1}}{A_Y^0}\Big)(A_Y^0-A_X^0)+p_{Y1}\,,$$

$$=\Big[p_{X0}p_{Y0}+p_{X0}\frac{(1-p_{Y0}-p_{Y1})}{A_Y^0}\Big](A_X^0-\epsilon)+\,,$$

$$\Big[\frac{(1-p_{X0}-p_{X1})(1-p_{Y0}-p_{Y1})}{2A_X^0A_Y^0}\Big](A_X^0-\epsilon)^2\,,$$

$$+\frac{(1-p_{Y0}-p_{Y1})}{A_Y^0}(A_Y^0-A_X^0)+p_{Y1}\,. \qquad (E.6)$$

Case 2: $\mathbf{A_X^0 > A_Y^0}$

$$p(Z\le 0)=\int_0^{A_X^0-\epsilon}\Big(p_{X0}+\frac{(1-p_{X0}-p_{X1})}{A_X^0}\,y\Big)p_Y(y)dy$$

$$+\int_{A_Y^0-\epsilon}^{A_X^0-\epsilon}f(y)p_Y(y)dy+\int_{A_X^0-\epsilon}^{\infty}f(y)p_Y(y)dy\,,$$

$$p(Z\le 0)=\Big[p_{X0}p_{Y0}+p_{X0}\frac{(1-p_{Y0}-p_{Y1})}{A_Y^0}\Big](A_Y^0-\epsilon)$$

$$+\Big[\frac{(1-p_{X0}-p_{X1})\,(1-p_{Y0}-p_{Y1})}{2A_x^0A_y^0}\Big](A_y^0-\epsilon)^2$$

$$+p_{X0}p_{Y1}+\frac{(1-p_{X0}-p_{X1})p_{Y1}A_Y^0}{A_X^0}\,. \qquad (E.7)$$

References

Abdul-Razaq, T.S., C.N. Potts, and L.N. VanWassenhove (1990), "A Survey of Algorithms for the Single, Machine Total Weighted Tardiness Scheduling Problem," *Discrete Applied Mathematics* **26**, 235–253.

Aho, A.V., J.E. Hopcroft, and J.D. Ullman (1974), *The Design and Analysis of Computer Algorithms,* (Addison Wesley, Reading, MA).

Apostol, Tom M. (1974), *Mathematical Analysis* (Addison Wesley, Reading, MA).

Baker, K.R. (1974) *Introduction to Sequencing and Scheduling* (John Wiley, New York, NY).

Berger, James O. (1980), *Statistical Decision Theory: Foundations, Concepts, and Methods* (Springer Verlag, New York, NY).

Bertsekas, Dimitri P., and J.N. Tsitsiklis (1989), *Parallel and Distributed Computation* (Prentice Hall, Englewood Cliffs, NJ).

Breiman, Leo (1968) *Probability* (Addison Wesley, Reading, MA).

Brent, R.P. (1973), *Algorithms of Minimization Without Derivatives* (Chapter 5) (Prentice Hall, Englewood Cliffs, NJ).

Castañon, David A., et al. (1989), *Advanced Weapon-Target Assignment Algorithms Program: Final Report* Report No. TR-440, (ALPHATEC, Inc., Burlington, MA).

Castañon, David A., et al. (1989), *Advanced Weapon-Target Assignment Algorithms Program: Algorithm Approaches Report* Report No. TR-428, (ALPHATEC, Inc., Burlington, MA).

Chung, Kai Lai (1974), *A Course in Probability* (Academic Press, New York, NY).

Coffman, E.G. (Ed.) (1976), *Computer and Job-Shop Scheduling* (John Wiley and Sons, New York, NY).

Conway, R.W., W.L. Maxwell, and L.W. Miller (1967), *Theory of Scheduling,* (Addison Wesley Reading, MA).

Corynen, G.C., and R.E. Glaser (1992), *Analysis and Optimization of Low Earth Orbit Communication Links*, Report UCRL-LR-109784, University of California, Lawrence Livermore National Laboratory, Livermore, CA.

Corynen, G.C. (1993), *Optimal Target Identification*, UCRL report in progress, University of California, Lawrence Livermore National Laboratory, Livermore, CA.

Daniels, Richard L. (1990), "A Multiple-Objective Approach to Resource Allocation in Single Machine Scheduling," *European Journal of Operational Research* **48**,221–241.

Devijver, P.A., and J. Kittler (1987), *Pattern Recognition Theory and Applications* (Springer Verlag, New York, NY).

Duda, R.O., and P.E. Hart (1973), *Pattern Classification and Scene Analysis* (John Wiley, New York, NY).

Dyer, M.E., and L.A. Wolsey (1990), "Formulating the Single Machine Sequencing Problem with Release Dates as a Mixed Integer Program," *Discrete Applied Mathematics* **26**, 255–270.

Eddy, W.F. (1977) "A New Convex Hull Algorithm for Planar Sets," *ACM Transactions on Mathematical Software*, Vol. 3, No.4, December.

Ferguson, Thomas S. (1967), *Mathematical Statistics: A Decision-Theoretic Approach* (Academic Press, New York, NY).

Flood, M.M. (1956), "The Traveling Salesman Problem," *Oper. Res.* 4, 61–75.

French, S. (1982), *Sequencing and Scheduling* (John Wiley, New York, NY).

Garey, M.R. and D.S. Johnson (1979), *Computers and Intractibility* (W.H. Freeman, San Francisco, CA).

Golden, B.L. and A.A. Assad (1986), "Vehicle Routing with Time-Window Constraints: Algorithmic Solutions," *American Journal of Mathematical Sciences*, **6**, Nos.3and 4, 251–428.

Holmes, R.B. and S.M. Rocklin (1990), *Analysis of Discrimination Sensors, Networks, and Measurement Allocation Strategies: An Overview* MIT Lincoln Laboratory Project Report Number DA-10, 15 October 1990, Lexington, MA.

Hosein, P.A., and M. Athans (1990), "An Asymptotic Result for the Multi-Stage Weapon-Target Allocation Problem," *Proc. 29th Conf. on Decision and Control*, Honolulu, Hawaii, Dec. 1990, 240–245.

Horowitz, E., and S. Sahni (1978), *Fundamentals of Computer Algorithms* (Computer Science Press, Rockville, MD).

Ibaraki, T., N. Katoh (1988), *Resource Allocation Problems* (MIT Press, Cambridge, MA).

Jain, Anil K. (1989), *Fundamentals of Digital Image Processing* (Prentice Hall, Englewood Cliffs, NJ).

Joachimsthaler, E.A., and A. Stam 1988), "Four Approaches to the Classification Problem in Discriminant Analysis: An Experimental Study," *Decision Sciences* **19** 322–333.

Koehler, G.J., and S.S. Evenguc (1990), "Minimizing Misclassifictions in Linear Discriminant Analysis", *Decision Sciences* **21** 63–85.

Kise, H., T. Ibaraki, and H. Mine (1978), "A Soluable Case of the One-Machine Scheduling Problem with Ready and Due Times," *Operations Research* **26**, No. 1 121–126.

Knuth, D.E. (1973), *The Art of Computer Programming, Volume 3: Sorting and Searching* (Addison-Wesley, Reading, MA).

Kuo, B.C. (1975), *Automatic Control Systems* (Prentice Hall, Englewood Cliffs, NJ).

Laporte, Gilbert (1992), "The Traveling Salesman Problem: An Overview of Exact and Approximate Algorithms", *European Journal of Operational Research* **59** 231–247.

Lawler, E.L. (1971a), "A Solvable Case of the Traveling Salesman Problem," *Math. Programming* **1**, 267–269.

Lawler, E.L., J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys (1985), *The Traveling Salesman Problem* (John Wiley and Sons, New York, NY).

Lin, S., and B.W. Kernighan (1973), "An Effective Heuristic for the Traveling Salesman Problem," **OR 21**, 498–516.

Loomis, L.H. and S. Sternberg (1968), *Advanced Calculus* (Addison Wesley Reading, MA).

Mealy, Gregory L., and G. Megaloudis (1989), *Fire Control Sensitivity Study* Final Report No. TR-5559-1, (The Analytic Sciences Corporation, Reading, MA).

Moore, J.M. (1968), "Sequencing n Jobs on One Machine to Minimize the Number of Tardy Jobs," *Management Science* **15**, 102-109.

Nemhauser, G.L., A.H.G. Rinnooy Kan, and M.J. Todd, (1989) *Optimization* (North-Holland, New York, NY).

Nemhauser, G.L., and L.A. Wolsey (1988), *Integer and Combinatorial Optimization* John Wiley and Sons, New York, NY).

Norback, J.P. and R.F. Love (1977), "Geometric Approaches to Solving the Traveling Salesman Problem," *Management Science* **23**, 1208–1223.

Papadimitriou, C.H. and K. Steiglitz (1982), *Combinatorial Optimization* (Prentice Hall, Englewood Cliffs, NJ).

Papoulis, A. (1965), *Probability, Random Variables, and Stochastic Processes* (McGraw Hill, New York, NY).

Papoulis, Athanasios (1977), *Signal Analysis* (McGraw Hill, New York, NY).

Press, W.H., B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling (1988), *Numerical Recipes in C* (Cambridge University Press, New York, NY).

Probst, Kevin B. (1992), Lawrence Livermore National Laboratory, Livermore, CA, private communications.

Reingold, E.M., J. Nievergelt, and N. Deo (1977), *Combinatorial Algorithms: Theory and Practice* (Prentice Hall, Englewood Cliffs, NJ).

Roberts, F.S. (1976), *Discrete Mathematical Models* (Prentice Hall, Englewood Cliffs, NJ).

Rocklin, S.M., and J.W. Tolleson (1986), *Discrimination Requirements for Multilayer Ballistic Missile Defense Systems*, MIT Lincoln Laboratory Technical Report No. 734, Lexington, MA.

Savelsberg, M. (1984), *Local Search in Routing Problems with Time Windows*, Report OS-R8409, Centre for Mathematics and Computer Science, Amsterdam, The Netherlands.

Seber, G.A.F. (1984), *Multivariate Observations* (John Wiley, New York, NY).

Sedgewick, R. (1988), *Algorithms* (Addison-Wesley, Reading, MA).

Smith, W.E. (1956), "Various Optimizers for Single-Stage Production," *Naval Research Logistics Quarterly* **3**, 59–66.

Solomon, M. (1986), "The Minimum Spanning Tree Problem with Time Window Constraints," *American Journal Math. Mgmt. Science*, **6**, 399–421.

Tsitsiklis, John N. (1992), "Special Cases of Traveling Salesman and Repairman Problems with Time Windows," *Networks* **22**, 263–282.

Vickson, R.G. (1980), "Two Single-Machine Sequencing Problems Including Controllable Job Processing Times," *AIEE Transactions* **12**, 258–262.

Vickson, R.G. (1980), "Choosing the Job Sequence and Processing Times to Minimize Total Processing Plus Flow Cost on a Single Machine," *Operations Research* **28**, 1155–1167.